PRIJS ƒ 7.25/Bfr. 135

Commodore INFO

EXTRA: Het grote Amiga-software overzicht

ONAFHANKELIJK BLAD VOOR COMMODORE GEBRUIKERS JAARGANG 5, NO. 1, JAN./FEBR. 1988



LISTINGS

- Checksum
- Adresboek
- Disk-bas
- Floppy 64
- Flash 64
- Orgel 128
- irqbasic
- Cardes 16
- Gokken C-16
- Kubus C-16
- Tape Disk C-16

Powercartridge gekraakt

Functie-analyse

LCN-logo

Vaste rubrieken

- Basic Miniatuurtjes
- Machinetaal
- Basic cursus
- Oud van Goudriaan
- Listings

Commodore Info

Verschijnt 8x per jaar
Jaargang 5, no. 1, januari 1988

Uitgave:
Sala Communications/SAC

Uitgever:
Drs. J. Taverne

Redactie:
Ir. L. Sala hoofdredacteur
J. Bodzinga adj. hoofdredacteur

J. Boers, drs. M. de Rooij, drs. H. Zoete, H. Smeenk, drs. U. Schuurmans, K. v.d. Vlies, R. Goudriaan, B. Munniksma, P.C. Broekhuizen.

Redactiesecretariaat:
R. van Zalingen

Strip: Bert Tier
Illustraties: Ben van Mierlo
Ymmot

Advertentie-exploitatie:
Ing. V. Sala
Weesperstraat 103
1018 VN Amsterdam
tel. 020-273198

Redactie adres:
Postbus 112
1260 AC Blaricum
tel. 02152-65695

Listingtelefoon:
(ma: 17.00-21.00) 02155-25162

Abonnementen en administratie:
Postbus 5570
1007 AN Amsterdam
tel. 020-273198

Vragen betreffende abonnementen ontvangen wij bij voorkeur schriftelijk, met meesturen van het omslagetiket.

Abonnement:
Voor 8 nummers ƒ 47,50 of BFR. 950 per jaar. Betaling op giro 1585491 t.n.v. SAC/Commodore-Info.
Oude nummers kunt U alleen krijgen bij vooruitbetaling van ƒ 6,75 op de bovenstaande rekening.
Ook telefonische opgave voor een abonnement is mogelijk. Bel GRATIS 06-02242222 (teleservice), elke dag tot 20.20 uur (dus ook in het weekend). België: 115555, dagelijks tot 22.00 uur. Deze telefoonnummers zijn alleen bedoeld voor opgave van NIEUWE abonnementen.

Druk: Verweij, Mijdrecht
NDB, Zoeterwoude

Distributie:
In Nederland: Betapress, Gilze
In België: AMP, Brussel

ISSN: 0169-3085

Inhoud van dit nummer



Redactioneel

Met grote verwachtingen ging ik begin januari naar de CES tentoonstelling in Las Vegas, waar Commodore traditioneel haar nieuws voor het komende jaar laat zien. Maar helaas hadden zowel Commodore als aartsconcurrent Atari besloten dit jaar deze beurs voor consumentenelectronica niet te gebruiken. Dat is jammer, maar ook weer een extra signaal, dat de dagen van de huiscomputer langzamerhand aan het aflopen zijn. Er worden nog wel veel computers verkocht, maar het image van de Amiga en de PC is wat serieuzer geworden. Men wil niet meer in de sfeer van de spelletjesmachine zitten, die markt is duidelijk overgelaten aan bijvoorbeeld Nintendo, dat 3 miljoen spelsystemen in de VS heeft verkocht en ondertussen bijna één miljard dollar omzet.

Natuurlijk waren er wel nieuwe dingen te zien in Las Vegas, waaronder nogal wat goede Amiga software, maar in het algemeen was de trend wat teleurstellend qua computers. Vendex (pacific), met Maurice de Hond nu officieel als president, was een van de weinige actieve bedrijven op de tentoonstelling. Verder waren in het kader van het Home Office de Personal Fax en ook Home Automation systemen van o.a. NEC en Mitsubishi de nieuwe trends in consumenten electronica. De winkeliers op audio en videogebied gaan de komende jaren waarschijnlijk hun computerhoek opdoeken en volzetten met dat soort apparatuur, zo blijft er altijd wel wat nieuws te verkopen. Voor het computernieuws hoeven we volgend jaar niet meer naar deze beurs, maar naar de Comdex computershow.
Luc Sala

De Power Cartridge van KCS is een veelzijdige hulp voor mensen die niet genoeg hebben aan een kale C-64. Het kastje maakt het onder andere mogelijk om sneller te laden en te saven, schermen te printen en gemakkelijker te programmeren in Basic en machinetaal. Je kunt er verder bijna alle programma's mee 'kraken', dat wil zeggen: je kunt ze onderbreken, eventueel saven of bekijken, en later weer vervolgen. De Power Cartridge is nu zelf gekraakt.

De Power Cartridge gekraakt

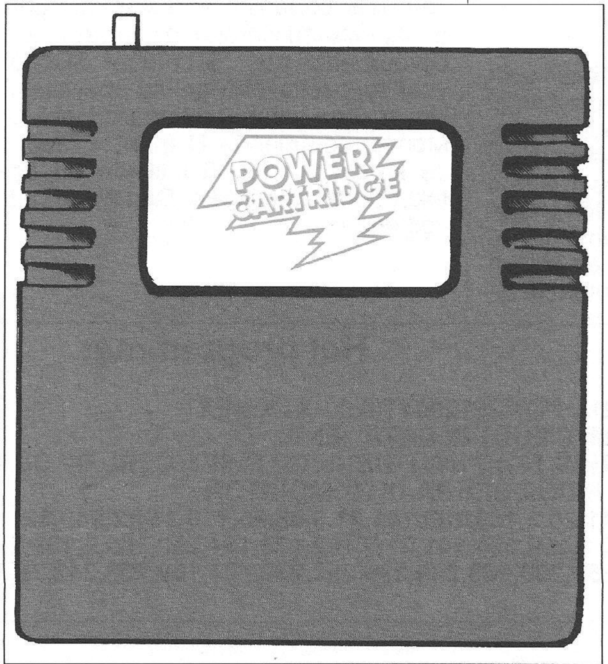
Wie wil leren over machinetaal, moet een cursus volgen of boeken bestuderen. Maar dat is nog niet genoeg, je moet ook nog ervaring opdoen. Je kunt zelf het een en ander proberen, maar het helpt vaak om het werk van professionele programmeurs te bekijken. Zo kun je veel leren over programmeertechnieken en over je computer. De ideale leerstof is natuurlijk een cartridge: nooit laadproblemen of wachttijden. De Power Cartridge bevat veelzijdige software, voor Basic-uitbreidingen, printer- en disk-sturing, geluid, sprites enzovoorts.

Het probleem was alleen: hoe kom je 'in' de cartridge? Welnu, als je dit artikel leest, zal het je wel duidelijk worden . De uitleg is soms vrij technisch, en kan eigenlijk alleen begrepen worden door mensen die al wat van de C-64 en van machinetaal afweten. Uiteraard wordt wel een poging gedaan om alles zo eenvoudig mogelijk te houden, maar soms ontkom je er niet aan een aantal technische, soms ingewikkelde begrippen te gebruiken. Daarna volgt de beschrijving van een korte speurtocht door de software in de cartridge.

Kopiëren naar RAM

De ROM van de Power Cartridge is aanwezig in het geheugen van de C-64 zodra de ingebouwde machinetaal-monitor gestart is met de opdracht MONITOR, en er verder geen programma lijkt te draaien. En daar gaat het om: het moet lijken alsof er geen programma loopt. Maar er zijn 'programma's' die bijna continu draaien, en dat zijn de interrupt-handlers. Dat zijn stukjes software die voor allerlei dingen zorg dragen waarbij regelmaat gewenst is, zoals het knipperen van de cursor, het bijhouden van de klok, en het lezen van het toetsenbord.

De vector ofwel pointer voor de IRQ (interrupt request = verzoek tot onderbreking) bevindt zich in de adressen 788 en 789 ($0314 en $0315) en kan gemakkelijk veranderd worden, zodat er een zelfgeschreven stukje software toegevoegd kan worden aan de standaard interrupt-handler.



De Power Cartridge

Programma

Het programma dat op het eind van dit artikel beschreven wordt zorgt ervoor dat de interrupt-pointer wijst naar een stuk machinetaal dat het geheugen van 32768 tot 40959 ($8000 tot $9FFF) op zichzelf kan kopiëren. Hierbij wordt het principe gehanteerd dat geldt bij het kopiëren van de Basic-ROM en de Kernal-ROM in de schaduw-RAM.
Het kopiëren gebeurt pas als de gebruiker dat zelf wil, en dat wordt aangegeven door op de SHIFT, CTRL of C=-toets te drukken. Tegelijkertijd wordt dan ook de pointer teruggezet, zodat hij naar het 'standaard' IRQ-adres wijst, om te voorkomen dat er elke keer weer gekopiëerd wordt, als je een van de aangegeven toesten indrukt.

Monitor niet verlaten

Het is voor het kopiëren van essentieel belang om de monitor niet te verlaten voor je een van de aangegeven toetsen hebt ingedrukt, want anders kopieert de computer alleen maar de RAM naar zichzelf.
Na het kopiëren kun je met M8000 8080 al vast even kijken of alles ook inderdaad gelukt. Wanneer je dan op kleine letters overschakelt, moet je ergens rechtboven CBM80 zien staan, de code die aan de rest van de computer meedeelt dat er een zelfstartende cartridge is. Is dat niet het geval, dan heb je waarschijnlijk ergens een fout gemaakt.

BASIC-variabelen

Onthoud overigens dat Basic-programma's die variabelen gebruiken of berekeningen uitvoeren hun gegevens opslaan in het geheugen op adres 40959 en lager, en dat is dus precies op de plek waar de ROM heen gekopiëerd is. De gegevens uit de ROM zullen verloren gaan en blijven als je variabelen gebruikt, totdat je alles opnieuw kopiëert. Dit kun je voorkomen door voor het kopiëren de volgende opdrachten te geven: POKE 56,128:CLR (return). Die zorgt ervoor dat het hoogste adres dat de Basic gebruikt, op 128x256 = 32768 komt te liggen, en dat er geen variabelen blijven bestaan in het oude gebied. Je kunt de bovenstaande opdrachten natuurlijk opnemen in het programma.

Verschillend

Bij gebruik van verschillende exemplaren viel op dat er tenminste twee verschillende versies van de Power Cartridge zijn. Ze zijn gemakkelijk van elkaar te onderscheiden: ze verschillen in de werking van functietoets F6. Bij de ene versie krijg je DSAVE" te zien, bij de andere LOAD"",2 gevolgd door een automatische return. Intern zijn ze ook enigszins anders, zo verschillen de meeste adressen enkele bytes.

Het zoeken

Als het kopiëren achter de rug is, volgt het zoeken naar interessante programma-gedeelten. Door de verschillende versies zul je de adressen die je hier aantreft moeten interpreteren als "omstreeks adres xxxx", en daar vlak voor of na zul je dan de betreffende machinetaal of tekst kunnen terugvinden.

Speurtocht

Op de speurtocht kun je het best zoeken naar bepaalde adressen die gebruikt moeten worden in de cartridge, bijvoorbeeld naar $D400 of $D000 voor respectievelijk geluid of sprites. Het zoeken gebeurt uiteraard met het Monitor-commando H (hunt = jaag). Je kunt ook M8000 intikken en de tekst door laten lopen. Dan zie je het volgende:

- 1) vanaf $8260: de Power Cartridge Basic-uitbreidingen;
- 2) vanaf $8438: de teksten die bij de functietoetsen horen;
- 3) vanaf $8530: de tekst 'backup', waarschijnlijk staat hier de standaard-naam voor bij een backup-tape of -disk zonder voorafgaande ILOAD"naam";
- 4) vanaf $9B28: de foutmeldingen van het monitor-programma, de tekst "*** powermon 2.0 ***", de te gebruiken een-letter- commando's van het monitor-programma, en de standaard register-regel, voor als je R intikt of de monitor binnenkomt.

Bij gewoon zoeken met H kun je het volgende proberen: H8000 A000 00 D4 (return), dit zoekt naar opdrachten die het eerste adres van de SID-chip gebruiken. Dit zal - allemaal bij deze versie van de cartridge - opleveren: 9BE0. Nu moet je dus daar ergens zoeken. Probeer daarom D9BD0 9BF0, en dan zie je een aantal JSR-opdrachten, gevolgd door een regel met: 9BDD LDA #$00. Doe dan: J9BDD (of het adres dat je zelf gevonden hebt voor de regel met LDA #$00), en je hoort de toon die je normaal alleen hoort als je in de monitor een fout maakt, en dat terwijl alles juist goed ging!

De hele cartridge doorzoeken

Op deze manier kun je de hele cartridge doorzoeken. Uiteraard kun je ook subroutines, zoals bijvoorbeeld voor het uitprinten van een grafisch scherm, eruit lichten en voor eigen gebruik aanwenden, eventueel met wijzigingen, en zonder de Power Cartridge. Bedenk wel dat er copyright op de software zit, dus probeer het maar niet te verkopen. Veel plezier bij het zoeken!

Peter Broekhuizen

Het programma:

```
10  FORT=0TO52:READA:POKE51200+T,A:X=X+A:NEXT
20  IFX<>7329THENPRINT"FOUT IN DATA":END
30  SYS51200:PRINTCHR$(147):PRINT"DRUK OP SHIFT, CTRL OF C=, DAARNA"
40  PRINT"IS DE CARTRIDGE GEKRAAKT!":MONITOR
50  DATA120,169,13,141,20,3,169,200,141,21,3,88,96,173,141,2,240,32
60  DATA169,49,141,20,3,169,234,141,21,3,162,128,134,252,160,0,132,251
70  DATA177,251,145,251,200,208,249,230,252,232,224,160,208,242,76,49,234
```

SOFT WIR WAR

DOOR BERT TIER.



MM.V LUC SALA.

Het invoeren van functies in een programma, is in de meeste gevallen een onmogelijke klus. De vreemdste capriolen worden soms uitgehaald om de functies toch in het programma in te voeren. Peter Heinckiens laat zien hoe een mogelijke oplossing voor dit probleem kan worden gevonden.

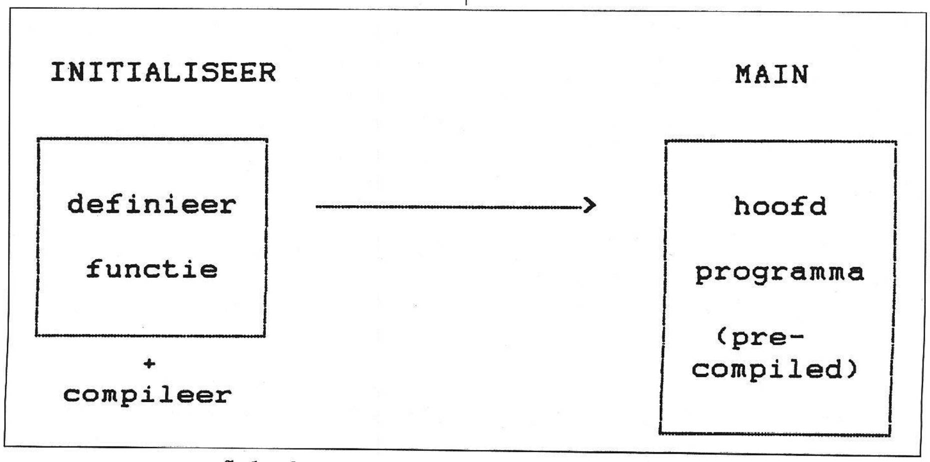
Sub-routines als praktische oplossing

Functie-analyse (1)

Iedereen die wel eens geprobeerd heeft een programma te schrijven waarbij de gebruiker een bepaalde formule moest kunnen ingeven, zal zich hebben afgevraagd hoe de computer zo kan worden geïnstrueerd, dat die formule ook begrepen wordt. De meeste programmeertalen laten immers niet toe dat een functie ingegeven wordt terwijl het programma loopt.

De meest rare trucs worden soms uitgehaald om toch functies te kunnen invoeren: men gaat het programma onderbreken, om de functie in de listing te plaatsen, of men probeert de formule rechtstreeks vanuit het programma in de listing POKEn (in BASIC).
Het probleem met deze methodes is echter dat het programma ten eerste niet meer overdraagbaar is naar andere machines, en dat men tevens problemen krijgt met het compileren van het programma naar machinecode. De programmalisting is immers verdwenen, en een formule plaatsen in een listing die niet bestaat, dát is pas echt moeilijk.
Een mogelijke oplossing is gebruik te maken van een sub-programma waarin de functie gedefinieerd wordt, en dit programma dan het hoofdprogramma te laten aanroepen(zie afb.1)

Maar ook dit is niet steeds mogelijk. Een voorbeeld hiervan is een spreadsheet (bv. LOTUS 1-2-3). Een dergelijk programma verdeelt het beeldscherm in verschillende velden, waartussen we bepaalde relaties kunnen definiëren.
Een spreadsheet kan er bijvoorbeeld als volgt uitzien :



afb.1 schematische voorstelling sub-programma

	A	B	C	D
1	getal1			
2	getal2			
3	som	<B1+B2>		
4	verschil	<B1-B2>		
5				
6				

Als we nu een getal in veld B1 invoeren, stel 5, en in B2 bijvoorbeeld het getal 6 plaatsen, dan komt in B3 de waarde 11 en in B4 de waarde -1 te staan :

	A	B	C	D
1	getal1	5		
2	getal2	6		
3	som	11		
4	verschil	-1		
5				
6				

De computer voert dus de in B3 en B4 voorgeschreven bewerkingen uit op de getallen.
Het is wel duidelijk dat die formules ingegeven moeten worden terwijl het programma loopt. Dus moeten we de computer in staat stellen ze te onderzoeken.

Een voorbeeld

Laten we eerst een vereenvoudigde versie van ons probleem bekijken: we gaan een programma schrijven dat een expressie ontleedt waarin de vier hoofdbewerkingen gebruikt mogen worden:

+, -, *, / .

Beschouw het volgende voorbeeld :

5 + 6 * 3 - 2 * (8 - 1)

We zouden deze uitdrukking nu natuurlijk onmiddellijk kunnen evalueren, maar indien dit verschillende keren moet gebeuren (wat het geval is als we een parameter gebruiken), kunnen we ze beter eerst onder een andere vorm schrijven: de *inverse-poolse* notatie, ook de 'postfix-notatie' genoemd. (Het is ook deze notatie die door HP-rekenmachines gebruikt wordt). In deze schrijfwijze wordt de bewerking achter zijn twee operandi geschreven. De uitdrukking "5 + 6" wordt in de postfix-notatie dus als "5 6 +" genoteerd. Bijgevolg dient ons voorbeeld als volgt herschreven te worden:

5 6 3 * + 2 8 1 - * -

Deze formule zegt ons nu dat we eerst 6 met 3 moeten vermenigvuldigen, en dan dit resultaat bij 5 optellen.
We noemen deze som SOM1. Vervolgens berekenen we 8-1, vermenigvuldigen dit met 2, en trekken we dit af van SOM1.
Maar, hoe slaan we deze formule nu op in een computer? Geen enkele programmeertaal laat 'mixed typing' toe, dit wil zeggen dat we geen getallen en lettertekens in dezelfde datastructuur mogen opslaan. Een mogelijke oplossing: stel iedere bewerking voor door een getal. Dit betekent echter dat we niet langer alle cijfers als cijfers zullen kunnen gebruiken. Dit is geen probleem wanneer we enkel positieve getallen gebruiken, we kunnen dan de bewerkingen voorstellen door negatieve.
Meestal zullen we echter ook getallen kleiner dan nul nodig hebben, en zullen we verplicht zijn een andere oplossing te zoeken. Maar, niet getreurd, we moeten de oplossing niet zó ver gaan zoeken: gebruik gewoon twee stacks: één voor de getallen en één voor de bewerkingen. Maar nu weten we niet meer in welke volgorde de elementen komen, bijgevolg zullen we een derde stack (de next-stack) nodig hebben die aangeeft welk soort element we krijgen: een operator of een getal. Onze stack kan er dan als volgt uitzien:

```
numbpointer= ^NUMBSTACK;
operpointer = ^OPERSTACK;
nextpointer = ^NEXTSTACK;

numbstack   = RECORD

              value    : INTEGER;
              link     : NUMBPOINTER;

      END (* numbstack *)
```

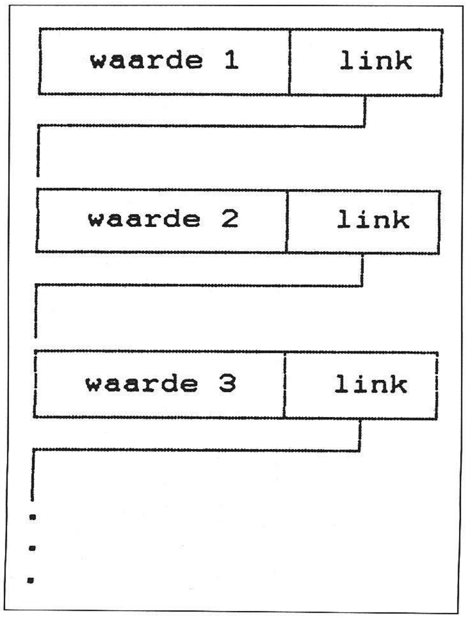
Analoog voor OPERSTACK en NEXTSTACK.

```
STACK = RECORD

    number   : NUMBPOINTER;
    operator : OPERPOINTER;
    next     : NEXTPOINTER;

    END (* stack *)
```

Merk op dat we gebruik gemaakt hebben van dynamische variabelen.
Een dynamische variabele is in feite een adres van een geheugenplaats waar een bepaalde variabele in is opgeslagen. Als samen met die variabele een wijzer (link) naar een volgend adres wordt gestockeerd, dan kunnen we een 'ketting' van variabelen vormen. Dit is schematisch weergegeven in afb.2.



afb.2 een 'variabele ketting'

Zonder gebruik te maken van dynamische variabelen, kunnen we de stack als volgt definiëren:

```
numbstack = array [1..30] of integer,-
operstack = array [1..30] of operand,-
nextstack = array [1..60] of possibility
```

Nu we de datastructuren die nodig zijn voor de stack, ontworpen hebben, kunnen we ons toespitsen op het evalueren van onze expressie. Beschouwen we ons voorbeeld: de stack ziet er als volgt uit:

number	operator	next
5	*	number
6	+	operator
3	-	operator
2	*	number
8	-	number
1		number
		operator
		operator
		operator

Bemerk dat de eerste twee elementen steeds getallen moeten zijn, bijgevolg zijn ze niet vermeld in de next-stack.
Om het algoritme te ontwerpen, zullen we ons in de plaats van de computer stellen:

° eerst nemen we een getal uit de number-stack (x1).
° het tweede element moet ook een getal zijn, dus haal het ook van de number-stack (x2).
° naar gelang de next-stack, zal het volgende element een getal (a) of een bewerking (b) zijn.

(a) een getal:

evalueer de formule vanaf x2, d.w.z.:
schuif x1 op een stack;
plaats x2 in x1;
doe de zelfde routine met de nieuwe x1, maar begin op (ii) }
goto (iii)

(b) een bewerking:

noem de operator b;
execute (x1, x2, b)

Als we het algoritme simpel willen houden, maken we gebruik van recursie (een routine roept zichzelf aan): we hoeven dan x1 niet meer op een stack te schuiven, daar hier automa-

tisch voor gezorgd wordt door de compiler. Opgelet: BASIC kan geen recursieve procedures verwerken. BASIC-programmeurs zullen dus een oplossing moeten zoeken om de recursie te omzeilen. De meer gestructureerde talen zoals PASCAL en C kunnen dit echter wel aan. Voor dit artikel heb ik gebruik gemaakt van PASCAL, maar het vertalen naar andere programmeertalen zou niet teveel problemen mogen geven. Ons programma (in pseudo-code) ziet er nu als volgt uit:

(1) Main

```
haal eerste getal (x1) van de number-stack;
WHILE er nog elementen zijn DO
      x1:= evaluate (x1)
      (bereken de cyclus vanaf x1en
      plaats het resultaat in x1 )
END.
```

(2) FUNCTION evaluate (x1)

```
BEGIN
haal x2 van de number-stack;
1:haal next van de next-stack;
CASE next OF
operator:   b:= operator;
      evaluate:= execute (x1, x2, b);
number:     x2:= evaluate (x2);
      GOTO 1
END.
```

(3) de functie EXECUTE berekent de waarde van de bewerking

```
      x1 b x2

bijvoorbeeld:
      x1 = 2
      x2 = 3
      b = +

=> execute (x1, x2, b) = 2 + 3 = 5.
```

Parameters

Natuurlijk is hiermee lang niet alles in orde: we moeten nog steeds een routine schrijven om de originele uitdrukking te vertalen naar de postfix-notatie. We zullen dit verderop behandelen. Laten we nu eens kijken hoe we onze opgave een beetje kunnen uitbreiden: we zullen ook een parameter toelaten. Dit betekent dat de computer nu drie zaken kan verwachten: een getal, een bewerking ofwel een parameter. Zoiets kan redelijk eenvoudig opgevangen worden door de next-stack.

(1) Main

```
CASE next-stack OF
      number: haal eerste getal (x1)
      van de numberstack;
      parameter:x1:= value;
END (* case *)

WHILE er nog elementen zijn DO
      BEGIN
       haal 'next' van de next-stack;
       x1:= evaluate (x1, next)
      END
END.
```

(2) FUNCTION evaluate (x1, next)

```
BEGIN
      CASE next OF
       number:haal x2 van de
       number-stack;
       parameter:x2:= value;
      END (* case *)

1:haal 'next' van de next-stack;

      CASE next OF
       operator:   b:= operator;
       evaluate:= execute (x1, x2, b);
       parameter,
       number: x2:= evaluate
       (x2, next);
       GOTO 1
      END (* case *)
END.
```

Zoals u ziet is er niet veel veranderd. Merk echter wel op dat de eerste twee elementen van de ingegeven functie niet noodzakelijk getallen meer zijn: het kunnen ook parameters zijn. Bijgevolg dienen ze nu wel opgenomen te worden in de next-stack. (Dit toont duidelijk aan dat het, met het oog op de uitbreidbaarheid van een programma, niet steeds verstandig is om teveel speciale trucjes te gebruiken).

Uitbreiding

Als we nu naast de vier hoofdbewerkingen, ook nog andere bewerkingen willen invoeren, dan kan dit door de functie 'execute' een beetje uit te breiden. Het wordt echter moeilijker, als we ook functies als sinus en cosinus willen onderzoeken. Om te beginnen zullen we de next-stack weer moeten uitbreiden: hij moet ons nu ook informeren als er een functie aankomt. Die functies moeten ook opgeslagen worden, daarom is er de behoefte aan nog een stack: de functie-stack. In ons algoritme blijft het MAIN-gedeelte onveranderd: het eerste element kan geen functie zijn. De module 'evaluate' ziet er nu als volgt uit:

(1) Main

```
CASE next-stack OF
      number  : haal x1 van de
      number-stack;
      parameter: x1:= value;
END (* case *)

WHILE er nog elementen zijn DO
      BEGIN
      haal 'next' van de next-stack;
      x1:= evaluate (x1, next)
      END
END.
```

(2) FUNCTION evaluate (x1, next)

```
BEGIN
      CASE next OF
       number : haal x2 van de
       number-stack;
       parameter: x2:= value;
       function: haal functie (f) van
       de function-stack;
       evaluate:= funcexec (f, x1);
      END (* case *)

      IF (nog niet klaar) and (next
      function) THEN
       REPEAT
       haal next van de next-stack;
       CASE next OF
       operator :b:= operator;
       evaluate:= execute (x1, x2, b);
       parameter,
       number:x2:= evaluate (x2, next);
       function:haal functie (f) van
       de stack;
       x2:= funcexec (f, x2);
       END (* case *)
       UNTIL next = operator;
END.
```

(3) FUNCEXEC retourneert het resultaat van f(x1)

```
      bv:
      f:= sin
      x1:= 0

      geeft: sin (0) = 0
```

Enige uitleg is hier wel op zijn plaats. Zoals u al weet blijft de procedure MAIN onveranderd, FUNCEXEC is betrekkelijk eenvoudig, dus laat ons ons toespitsen op EVALUATE. Eerst wordt gekeken of 'next' een functie is, zo ja wordt het resultaat van die functie in x1 aan 'evaluate' toegewezen. De routine eindigt nu. Indien 'next' geen functie was (i.e. het was een getal of een parameter), dan komen we in het tweede deel van onze routine, waar we nu de mogelijkheid beschouwen dat we deze keer wel een functie zullen krijgen. Als dit het geval is, dan wordt deze door FUNCEXEC op dezelfde manier verwerkt als een bewerking door EXECUTE. Het enige verschil is dat na een operator de routine stopt, maar na een functie niet.
Laat ons dit alles illustreren aan de hand van volgend voorbeeld: we willen de onderstaande uitdrukking evalueren:

INC(5) + 3*xmet x = 2.

Eerst vertalen we dit naar de postfix-notatie:

5 INC 3 x * +

Onze stack ziet er als volgt uit:

Numb.	Oper.	Func.	Next
5	*	inc	number
3	+		function
			number
			parameter
			operator
			operator

afb.3

Nu roepen we de routine MAIN aan: Aangezien het eerste element een nummer is, wijzen we dit toe aan x1:

x1:= 5.

Er zijn nog elementen over (de next-stack is nog niet leeg), dus halen we 'next' van de next-stack en roepen 'evaluate' aan:

x1:= evaluate (5, function).

In EVALUATE gebeurt het volgende:

'Next' is een functie, dus halen we f van de stack en roepen FUNCEXEC aan:

f:= inc;
evaluate:= funcexec (inc, 5)

Hierdoor krijgt 'evaluate' de waarde 6.

Aangezien 'next' een functie is, eindigt de routine.

daardoor krijgt x1 de waarde 6:

x1:= 6.

Nu komen we aan het tweede deel van de MAIN-routine en aangezien er nog elementen over zijn, wordt het uitgevoerd. 'Next' wordt nu:

next:= number

en we krijgen voor x1:

x1:= evaluate (6, number).

Nogmaals roepen we EVALUATE aan:

next = number
=> haal x2 van de stack:
x2:= 3.

We zijn nog niet klaar (de next-stack is nog niet leeg), dus halen we 'next':

next:= parameter

Dit houdt in dat we nogmaals EVALUATE zullen moeten aanroepen. Let op: dit is een 'recursieve aanroeping'.

x2:= evaluate (3, parameter).

In deze cyclus ziet EVALUATE er als volgt uit:

x1 := 3
next:= parameter
=> x2:= value
=> x2:= 2

We zijn nog steeds niet klaar, dus haal 'next':

next:= operator

=> b:= times
evaluate:= execute (3, 2, times)
=> evaluate:= 6

Aangezien 'next' een bewerking is, eindigt de routine. X2 wordt:

x2:= 6.

'Next' is geen operator, bijgevolg dienen we de lus nogmaals te doorlopen:

next:= operator

=> b:= plus
evaluate:= execute (6, 6, plus)
=> evaluate:= 12

N is 'next' een bewerking, dus eindigt de routine. Zo krijgt x1 de waarde:

x1:= 12.

En de next-stack is leeg, dit betekent dat we klaar zijn.

Listings

Dit alles zou u nu in staat moeten stellen om uw eigen routines te schrijven, en bij wijze van voorbeeld heb ik er één bijgevoegd die gehele getallen (integers), parameters, de vier hoofdbewerkingen en de functies INC en DEC kan verwerken (listing 1). Op het eerste gezicht is deze routine erg lang uitgevallen, doch dit komt voornamelijk doordat er uitvoerig gebruik gemaakt is van witte ruimtes en procedures. Dit vergroot de leesbaarheid aanzienlijk. De TYPE-definities nodig voor deze en de volgende routine bevinden zich in listing 2.
In het volgende nummer zal Peter Heinckens verder ingaan op het omzetten van formules van infix- naar postfix-notatie.

LISTING 1: Pascal equivalent van het algoritme EVALUATE

```
FUNCTION evaluate  ( stack : POLISH;
                     value : INTEGER) : INTEGER;

(*
     EVALUATE 1.2          -        4 april 1987

     Auteur          :  Peter Heinckiens

     Doel            :  Deze module berekent de waarde van de mathematische
                        uitdrukking in de stack. Alle getallen moeten van
                        het type INTEGER zijn. De volgende bewerkingen zijn
                        toegelaten : +, -, *, div. De functies INC en DEC
                        zijn ook opgenomen. Het gebruik van een parameter
                        i.p.v. een getal is toegelaten.

     Aanroeping    :  y := evaluate (stack, value)
met : stack van het type POLISH

POLISH = RECORD
           numb     : NUMBPOINTER
           oper     : OPERPOINTER
           func     : FUNCPOINTER
           next     : NEXTPOINTER
         END

numb : wijst naar de getallen,

oper : wijst naar de bewerkingen :

          plus
          minus
          times
          slash

func : wijst naar de functies :

          inc
          dec

next : bepaalt het soort element :

          operator
          number
          parameter
          functie

'value' is de waarde van de parameter.

VAR x1        : INTEGER;
    element   : KIND;

FUNCTION getoperator : SYMBOL;
VAR reserve : OPERPOINTER;
BEGIN
  getoperator := stack.oper^.value;
  reserve     := stack.oper;
  stack.oper  := stack.oper^.link;
END;

FUNCTION getnumber : INTEGER;
VAR reserve : NUMBPOINTER;
BEGIN
  getnumber   := stack.numb^.value;
  reserve     := stack.numb;
  stack.numb := stack.numb^.link;
END;

FUNCTION getfunc : FUNCTIONS;
VAR reserve : FUNCPOINTER;
BEGIN
  getfunc    := stack.func^.value;
  reserve    := stack.func;
  stack.func := stack.func^.link;
END;

FUNCTION getnext : KIND;
VAR reserve : NEXTPOINTER;
BEGIN
  getnext    := stack.next^.value;
  reserve    := stack.next;
  stack.next := stack.next^.link;
END;

FUNCTION not_ready : BOOLEAN;
BEGIN
  not_ready := stack.next <> nil
END;

FUNCTION calculate ( x1       : INTEGER;
                     element : KIND ) : INTEGER;

VAR   x2  : INTEGER;
      b   : SYMBOL;
      f   : FUNCTIONS;
```

```
FUNCTION execute (x1, x2 : INTEGER;
                  b      : SYMBOL ) : INTEGER;

BEGIN

  CASE b OF

     plus    :  execute := x1 + x2;
     minus   :  execute := x1 - x2;
     times   :  execute := x1 * x2;
     slash   :  execute := x1 div x2;

  END;   (* case *)

END;


FUNCTION funcexec (f : FUNCTIONS;
                   x : INTEGER   ) : INTEGER;

BEGIN

  CASE f OF

    inc  :  funcexec := x+1;
    dec  :  funcexec := x-1;

  END;  (* case *)

END;


(* calulate *)

BEGIN

  CASE element OF

    number    :   x2 := getnumber;
    parameter :   x2 := value;
    functie   :   BEGIN   f  := getfunc;
                          x1 := funcexec (f, x1);
                          calculate := x1;
                  END;

  END;  (* case *)
```

```
   IF not_ready and (element <> functie) THEN

     REPEAT

       element := getnext;

       CASE element OF

         operator  :  BEGIN    b := getoperator;
                               calculate := execute (x1, x2, b)
                      END;

         number,
         parameter :  BEGIN    x2 := calculate (x2, element);
                      END;

         functie   :  BEGIN    f  := getfunc;
                               x2 := funcexec (f, x2);
                      END;

       END;  (* case *)

     UNTIL element = operator;

END;


(* evaluate *)

BEGIN

  CASE getnext OF
    number     :  x1 := getnumber;
    parameter  :  x1 := value;
  END;  (* case *)

  WHILE not_ready DO

    BEGIN
      element := getnext;
      x1 := calculate (x1, element);
    END;

  evaluate := x1;

END;
```

LISTING 2: TYPE-definities nodig voor de twee procedures

```
(*
  Type-definities nodig voor de modules 'reverse_polish' en 'evaluate'.
*)


TYPE  kind          = (operand, number, parameter, functie);
      symbol        = (plus, minus, times, slash);
      functions     = (inc, dec);
      numbpointer   = ^numbstack;
      operpointer   = ^operstack;
      funcpointer   = ^funcstack;
      nextpointer   = ^nextstack;
      polish        = RECORD
                        numb : NUMBPOINTER;
                        oper : OPERPOINTER;
                        func : FUNCPOINTER;
                        next : NEXTPOINTER;
                      END;
      numbstack     = RECORD
                        value : INTEGER;
                        link  : NUMBPOINTER;
                      END;
      operstack     = RECORD
                        value : SYMBOL;
                        link  : OPERPOINTER;
                      END;
      funcstack     = RECORD
                        value : FUNCTIONS;
                        link  : FUNCPOINTER;
                      END;
      nextstack     = RECORD
                        value : KIND;
                        link  : NEXTPOINTER;
                      END;

      formula_type = STRING [60];
```

Skyfox II

Hevige ruimtegevechten vinden plaats in deze tweede versie van Skyfox. Deze versie heeft een nieuw verhaal, betere grafische animatie, snellere aktie en het vliegen is sensationeler. De aktie vindt plaats in de uithoeken van het universum, waar de strijders van de Federatie het opnemen tegen de Xenomorphs.
Als speler ben je Federatie-piloot en je hebt de beschikking over een geavanceerd Skyfox II gevechtsvliegtuig.Het spel kent verschillende behendigheidsniveaus en tien mogelijke gevechtssituaties. De Skyfox II beschikt over een modern wapenarsenaal, zoals neutronenstralen, Photonbommen en anti-materie mijnen. De speler kan sneller dan het licht vliegen door gebruik te maken van wormgaten, dat zijn bijprodukten van zwarte gaten.
Inl: Electronic Arts, tel. Engeland 0753 49442

Epyx

De Amerikaanse software ontwikkelaar Epyx komt met drie spelletjes op de markt.
Street Sports Basketball onderscheidt zich van andere basketbalspellen doordat de speler het speelveld zelf kan kiezen: schoolplein, steeg, achtertuin of parkeerplaats. Elk veld heeft z'n voor- en nadelen. Ook kun je kiezen of je tegen de computer of een andere speler speelt. Vervolgens moet je je team samenstellen, waarbij je de keuze hebt uit tien spelers, elk met z'n eigenaardigheden. Zoals Ralph, snel en lenig, maar hij draagt altijd een pet die soms over z'n ogen glijdt, juist als hij een meesterschot wil afvuren. De spelers worden bewogen door middel van de joystick.
Het tweede spel is **Arctic Antics:Spy vs Spy III**, waarbij de spelers gestrand zijn op een ijsberg. Een dodelijke sneeuwstorm is op komst! Er is maar een uitweg en ze moeten alles in het werk stellen om in de vluchtraket te komen. Op de ijsberg worden ze dwars gezeten door een vijandelijke spion en de barre natuurlijke omstandigheden maken de vlucht gevaarlijk. Ze moeten dun ijs, diepe sneeuw en lange ijspegels ontwijken.De spelers hebben wel een aantal hulpmiddelen, zoals zagen, dynamiet, pikhouwelen en ontstekers. Met de spion moet nog een robbertje gevochten worden voordat de raket afgevuurd kan worden.
Het derde spel is een variatie op de bekende grotten- en spelonkenspellen. **Boulder Dash Construction Kit** biedt echter meer. Met de Construction Kit kun je namelijk zelf de spellen vorm geven. Je kunt ze, door het toevoegen van allerlei elementen, zo lang en moeilijk maken als je maar wilt. Rotsen, groeiende muren en enge beesten kun je invoeren. Doel van het spel is om zo veel mogelijk diamanten te verzamelen, die in de grotten verstopt zijn. Inl: Epyx, tel V.S.(415)366-0606

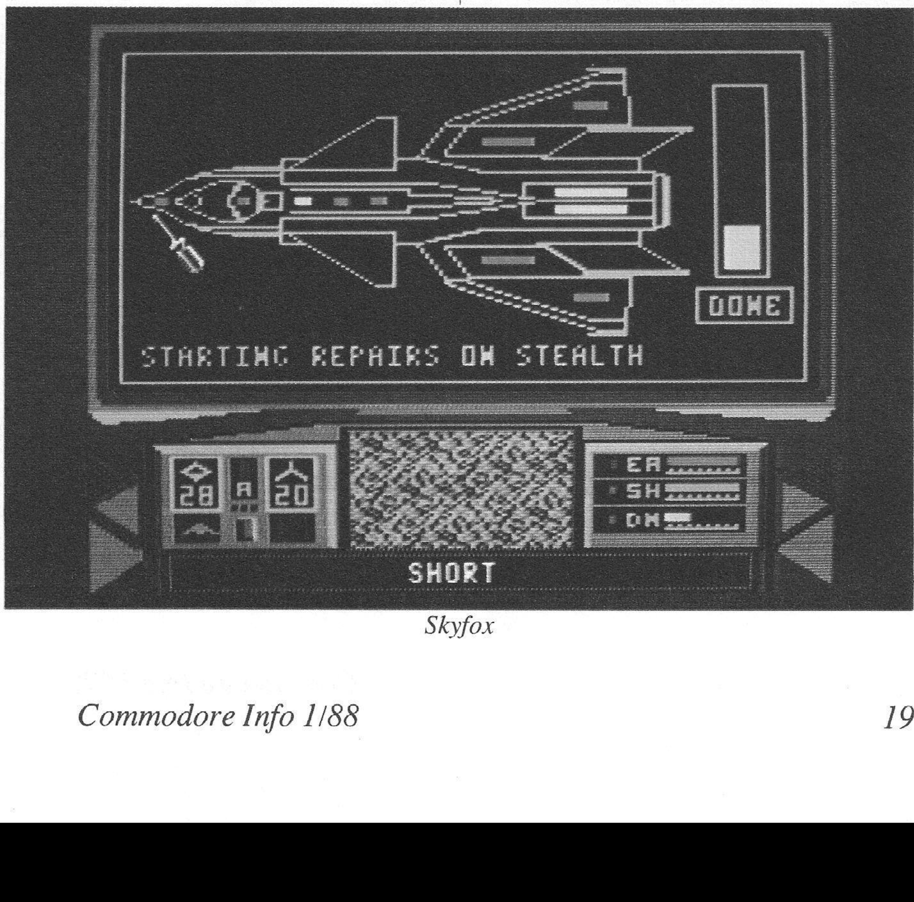
Per 1 december jl. heeft Electronic Arts de Europese distributie van Martech software op zich genomen. Martech is vooral bekend door spellen als Nigel Mansell's Grand Prix, Catch 23, Slaine en Mega-Apocalypse.Inl: Electronic Arts, tel. Engeland 0753 49442

Easy Working

Spinnaker Software in de VS heeft drie Easy Working programma's voor o.a. de Commodore 64/128 op de markt gebracht voor rond de tien dollar per stuk. De programma's zijn 'The writer', 'The Filer' en 'The Planner'. Ze kunnen in combinatie met elkaar gebruikt worden. De programma's werken met drop-down menu's. Elk beschikbaar commando wordt in het menu weergegeven. 'The writer' heeft de meest gangbare tekstverwerkingsopties. 'The Filer' maakt het opslaan, selekteren en opvragen van informatie eenvoudig. Het is geschikt voor onder meer mailing lists, inventarissen, lidmaatschapslijsten. 'The Planner' is een spreadsheet en is te gebruiken voor bijvoorbeeld budgettering, financiele administratie en belastingberekening.
Dezelfde firma brengt een '7 in 1' softwarepakket uit. Het pakket bevat een desktop organizer, outliner, wordprocessor, spreadsheet met Lotus 1- 2-3 export en import, database, business graphics en communications. Alle zeven toepassingen gebruiken elkaars gegevensbestanden. U kunt zo bijvoorbeeld met de spreadsheet cijfers uit de data base analyseren, de gegevens grafisch weergeven en in een brief verwerken. De prijs van het pakket ligt rond de 65 dollar. Inl: Spinnaker Software tel: U.S.A. (617)494-1200

Not a penny more!

Domark brengt een computerspel uit van het boek Not a penny more, not a penny less! Het is een verhaal van internationale fraude en intriges. De hoofdpersoon verliest een groot deel van z'n in aandelen belegde fortuin aan een oplichter. De aandelen blijken waardeloos en hij zint op wraak. Samen met andere bedrogen investeerders ontwikkelt hij een plan om het geld terug te krijgen. Een vernuftig web van intriges wordt gesponnen rond de oplichter. In de computerver-



Skyfox

sie van het boek is de speler de hoofdpersoon. Je moet het team van amateur-oplichters samenstellen en een serie van trucs en oplichterijen afwikkelen voordat het verloren geld terug is. Het leuke van dit spel is dat het scenario van het boek nauwkeurig gevolgd wordt. Op de talloze lokaties die je in het spel aandoet liggen puzzles die je eerst moet oplossen voor je verder kunt. Het pakket bevat het spel, instrukties en een kopie van het originele boek.
Inl: Domark, tel. Engeland 01-947 5622.

Tetris

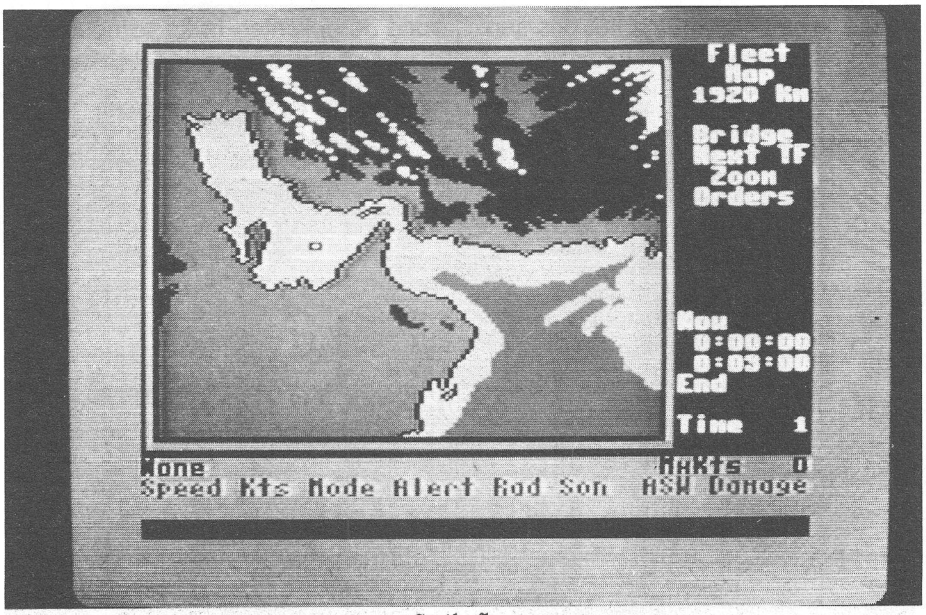
Een simpel spel van Mirrorsoft, maar je schijnt er behoorlijk aan verslaafd te kunnen raken. Een serie verschillende vormen verschijnt bovenaan het scherm. Met behulp van het keyboard of de joystick moet je die vormen zo manipuleren, dat ze aan de onderkant van het scherm tot lijnen samengevoegd worden. Hoe sneller je de lijnen kunt maken, hoe sneller de vormen verschijnen. Verkrijgbaar voor o.a. Commodore 64/128 (tape en disk) en voor Amiga.
Inl:MirrorSoft, tel. Engeland 01-377 4837

Strike fleet

Een aktueel spel is Strike fleet van Electronic Arts. Het is een simulatie van zeegevechten waarbij zowel strategie als harde aktie om de hoek komen kijken. Als commandant van een oorlogsvloot, uitgerust met moderne wapens, moet je tien gevaarlijke opdrachten uitvoeren. Zo moet je bijvoorbeeld alle neutrale schepen in de Perzische Golf verdedigen en supertankers escorteren.Ook zijn enkele Wereld Oorlog III scenario's opgenomen. Je kunt een scenario per keer nemen, of meerdere tegelijk en ook kun je de grootte en uitrusting van de vloot kiezen. Het doel is om alle opdrachten uit te voeren met zo min mogelijk verlies van bemanning en materieel. Als je daarin slaagt, ontloop je een militair tribunaal en wordt je bevorderd. Inl: Electronic Arts, tel. Engeland 0753 49442

Andy Capp

Ook van Mirrorsoft komt Andy Capp, bij ons beter bekend als Linke Loetje. Het is vrijdagavond en de cheque van Andy's uitkering wordt vermist. Hoe kun je uitvinden wie hem achterover gedrukt heeft? Om daarvoor de juiste informatie van vrienden en kennissen te krijgen moet Andy ze paaien. En dat is juist niet zijn sterkste punt. Dus Andy heeft geen andere keus dan z'n laatste geld bij de paardenraces te vergokken, hopend op een winnend lot. In het spel kun je alle bekende overlevingstechnieken van Andy toepassen: vechten met de bookkeeper, de huisbaas een klap verkopen, bedelen bij de barkeeper, de portemonnee van je vrouw achterover drukken etc. Inl: Mirrorsoft, tel. Engeland 01-377 4837



Strikefleet

Stifflip

Een oer-engelse vlieger in WO-I outfit met monocle, diverse kartoonachtige personages, een comic-book game en een slechte oom vormen de ingrediënten van Binary Visions **Stifflip**. Graaf Stifflip is de held in dit jongensboek-adventure. De heldenda-den worden in een bananenrepubliek verricht. Kortom de ware sfeer van een comic book.
De plaatjes zijn in kartoonstijl getekend. In het midden van het scherm speelt de feitelijke actie zich af. Links staan de actiemogelijkheden (Chinwag, Fisticuff, Beetle-off, State of play en Change Batter) en rechts de beschikbare personages (Colonel R.G. Bargie, Professor Braindeath en miss Palmyra Primbottom) per spelpagina. De speler bestuurt het spel door links en rechts het gewenste ikoon te kiezen. Alle vier karakters spelen hun eigen bestuurbare rol in dit game.
Graaf Stifflip blijft altijd onverstoorbaar en hyper(engels)- beleefd. Kreten als "Now look here my good man" of "Nice bar you have here" verschijnen in gepaste tekstballonnen. Helaas haalt beleefdheid bij vijanden weinig uit en is het knokken geblazen voor de brave Stifflip. Gelukkig is ook het knokken overgoten met een sausje van echte onvervalste droge humor. Natuurlijk moet het vechten via de gentleregels gaan. De enige arcade-actie van Stiflipt verloopt via "Bif!" (Fisticuff) waarbij de speler een ikoon op een roterend doel dient te plaatsen en vervolgens op de vuurknop van de joystick drukt. Slagen onder de gordel brengen in deze gentlemans agreement boks-arcades minder punten op als het sportieve boven de gordel meppen.
Het verhaal doet er eigenlijk niet zo toe. Het is voornamelijk sfeertekenend. In de roaring twenties wordt het Britse wereldrijk bedreigd door de snode graaf Chamaleon. Deze schurk treedt alle herenregels met voeten en heeft een "Rubbertronic Ray" ontwikkeld waarmee hij, oh dear!, snorrebaarden slap maakt, stijve boorden naar hun grootje helpt en cricketballen kan afbuigen. Een echte ramp dus. In vier avonturen moet Sebastiaan Stifflip dit zaakje in de bananenrepubliek klaren. Zoals gezegd kunnen naast Stiflip ook de andere figuren gebruikt worden. verder zal de avonturier nog tal van kleurrijke personages ontmoeten. Stifflip moet het voornamelijk van de humoristische sfeer hebben.

LOGO is een echte programmeertaal die vooral voor het gebruik in het onderwijs ontworpen. In mei 1987 introduceerde de Stichting LOGOgroep Nijmegen een Nederlandse LOGO voor de Commodore 64; inmiddels is er ook een MS-DOS versie.

LCN LOGO

Een Nederlandse LOGO



Het nederlandse LOGO is met name bedoeld om leerlingen van het voortgezet onderwijs , maar ook de hoogste groepen van het basisonderwijs 'zelfontdekkend' te laten leren. Centraal hierbij staan hierbij doorgaans de spektakulaire grafische mogelijkheden van LOGO en het manipuleren van objecten, de sproken, zgn. 'schildpaddenklonen'.In dit artikel besteden we aandacht aan een aantal deze en andere typerende kenmerken van deze Nederlandse LOGO. LCN-LOGO wijkt in sommige opzichten namelijk nogal af van conventionele LOGO's.

'Leren met LOGO' was de titel van een onlangs aan de Katholieke Universiteit Nijmegen gehouden konferentie. Wat kinderen precies leren van LOGO bleek niet zo eenvoudig vast te stellen. Uit met name Amerikaans onderzoek blijkt dat kinderen die langere tijd met LOGO werken vaardigheden ontwikkelen die te maken hebben met het analyseren en planmatig aanpakken van taken. In vergelijking met leeftijdgenootjes gaan ze bij het oplossen van problemen systematischer en doelmatiger te werk. Daarnaast geven hun wiskundige en ruimtelijke inzichten vaak een forse voorsprong te zien.
In Nederland werd LOGO vijf jaar geleden geïntroduceerd door het LOGO Centrum Nederland, een initiatief van de Stichting LOGOgroep Nijmegen. In de Verenigde Staten was toen reeds sprake van een ware rage. Een bepaalde LOGO-versie werd daar zelfs tot tweemaal toe uitgeroepen tot beste edukatieve software van het jaar. Hier in Nederland ging het er allemaal iets rustiger aan toe. Weliswaar ging een aantal scholen LOGO gebruiken, de meeste bleven toch de leerlingen lastig vallen met het traditionele BASIC. Pas bij het beschikbaar komen van Nederlandstalige versies (vooral voor de MSX en Commodore) werd LOGO meer en meer een normaal verschijnsel op school, i.p.v. BASIC. Sedert kort bestaat er een heuse Nederlandse LOGO.

LCN LOGO

De LCN-LOGO is een geheel Nederlands produkt en komt uit Nijmegen. Het heeft enige jaren geduurd voordat het produkt verkoopgereed was. Dat lag hem voornamelijk in het feit dat in eerste instantie gemikt werd op een Commodore-versie.
De Commodore is nooit het voorbeeld geweest van een fraaie inwendige architectuur, en daarom duurde het project veel langer dan gepland. Bovendien moest de LCN-LOGO zoveel in zich hebben dat de geheugenruimte van de Commodore dit allemaal niet aankon. Vandaar dat de Commodore-versie is ondergebracht in een insteekmodule. Het aantal PC's op scholen nam zo toe dat onmiddellijk werd begonnen met een MS-DOS-versie. Een nadeel hierbij is dat men voor LCN LOGO een CGA- of EGA-kaart nodig heeft: met een Hercules-kaart werkt het systeem niet. Voor machines die voldoen aan de NIVO-specifikaties is er dus geen vuiltje aan de lucht. Dankzij het bijgeleverde optie-file kan er zowel monochroom als in kleur gewerkt worden. Een ander nadeel t.o.v. de Commodore-versie is dat er geen mogelijkheden voor sproken zijn. Maar het produkt is dermate fraai geworden dat de nadelen niet opwegen tegen alle voordelen die LCN-LOGO biedt.

Menu's en schermen

Wat bij het starten van LCN LOGO meteen al opvalt is dat het beeldscherm in delen is opgesplitst. Het bovenste deel van het scherm is gereserveerd voor de editor, en het middelste deel voor tekst-uitvoer. De positie van de grens tussen de beide schermpjes is instelbaar. Helemaal onderaan bevindt zich één (gekleurde) regel die gebruikt wordt voor systeem- en foutmeldingen, en die daarom statusbalk genoemd wordt. Met een druk op de knop kan de gebruiker switchen naar het grafisch scherm, waarbij zich in het midden de bekende schildpad bevindt.
Een groot aantal handelingen verloopt op een menu-gestuurde manier. Via een funktietoets komt men terecht in

een overzichtelijk hoofdmenu. Men kan hier kiezen uit de opties 'LEZEN' (voor het inlezen van programma's of afbeeldingen), "SCHRIJVEN" (voor het bewaren of printen van programma's en afbeeldingen), 'DOS' (voor de kommunikatie met MS-DOS), 'HULP' en 'STOPPEN'. Kiest men 'HULP', dan komt men terecht in een 'hulphoofdmenu', van waaruit men allerhande informatie kan opvragen over het gebruik van het systeem (over de kommandotoetsen bijvoorbeeld). Op zich is een dergelijke hulpfaciliteit niets bijzonders. Bij LCN LOGO bevindt de inhoud van de hulpwindows zich echter in aparte bestanden die door de gebruiker naar believen kunnen worden aangepast en uitgebreid. Voor docenten biedt dit bijv. de mogelijkheid eigen LOGO-programma's te voorzien van makkelijk op te roepen kommentaar, uitleg en aanwijzingen. Zo'n programma hoeft overigens niet voor de leerling zichtbaar te zijn: men kan het beschermd wegschrijven. Het is dan weliswaar door derden te gebruiken, maar de tekst ervan is onzichtbaar.

Directe en indirecte modus

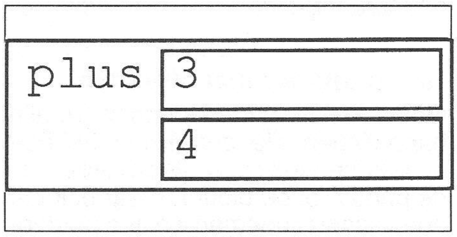
In bijna alle LOGO-versies bestaat er een duidelijk verschil tussen de kommando-toestand en de editor-toestand. In de kommando-toestand worden de opdrachten die de gebruiker intypt onmiddellijk uitgevoerd na <RETURN>. Men spreekt ook wel van 'directe modus'. In de edit-toestand worden opdrachten na <RETURN> niet meteen uitgevoerd, maar wordt er gewacht tot men op een speciale toets drukt, waarna alle opdrachten in de editor worden uitgevoerd. Op deze manier kan men procedures maken.

In LCN LOGO bestaat het onderscheid tussen directe en indirecte modus niet. Deze LOGO kent alleen een editor-toestand. Door de cursor als aanwijsinstrument te gebruiken kan men de betreffende opdrachten uit laten voeren. Men hoeft hiervoor de editor niet te verlaten. De LCN LOGO editor kombineert dus de voordelen van de kommando- en editor-toestand. Kommuniceren met LCN LOGO geschiedt uitsluitend via de editor.

De editor

Als elke taal heeft LOGO een eigen struktuur. Bij een ALS-opdracht hoort bijvoorbeeld altijd een DAN-gedeelte, enz. Deze struktuur wordt niet ondersteund door een konventionele LOGO-editor. Desondanks is ook daar sprake van een zekere struktuur. Maar deze behelst niet veel meer dan die welke men tegenkomt bij tekstverwerkers: woorden vormen zinnen en zinnen vormen paragrafen. Met andere woorden: conventionele versies van de taal zijn regel-georiënteerd.
Omdat een traditionele LOGO-editor de syntax van de taal niet ondersteunt, is het soms noodzakelijk om extra informatie toe te voegen. Deze bestaat bijvoorbeeld uit het plaatsen van haakjes om aan te geven wat bij wat hoort, of het toevoegen van bijzondere leestekens om aan te geven dat een bepaald woord een speciale betekenis heeft. Door deze noodzaak om extra informatie toe te voegen wordt de (vaak jonge) beginner niet zelden tot wanhoop gedreven. Wanneer de haakjes bijvoorbeeld ontbreken is de tekst vaak ambigu en wordt er een foutmelding gegeven of gebeurt er iets onverwachts.

De makers van LCN LOGO hebben daar iets op bedacht. Deze LOGO heeft slechts één syntax- of struktuurregel, en die wordt konsequent volgehouden. In LCN LOGO wordt namelijk de prefix-notatie gehanteerd, d.w.z. eerst schrijven we de naam van de opdracht (de naam van het "primitief" in LOGOjargon), pas dan volgen de eventuele argumenten. Is er sprake van meerdere argumenten, dan worden die zonder uitzondering onder elkaar geplaatst.
In de afbeelding is aangegeven hoe de gebruiker zo'n opdracht intypt. De editor helpt hem daarbij door de cursor na <RETURN> niet naar het begin van de regel te sturen, maar naar de positie precies onder het eerste argument (t.w. de plaats waar het tweede argument thuis hoort).
De afbeelding boven aan de pagina laat zien dat er in dit voorbeeld drie "vlakken" zijn: plus met zijn beide argumenten is een vlak. Maar die argumenten zijn zelf k weer vlakken. Deze vlakken worden zichtbaar wanneer men met de cursortoetsen door de tekst loopt. Daardoor is vooral bij grotere, samengestelde opdrachten de struktuur goed te doorzien.

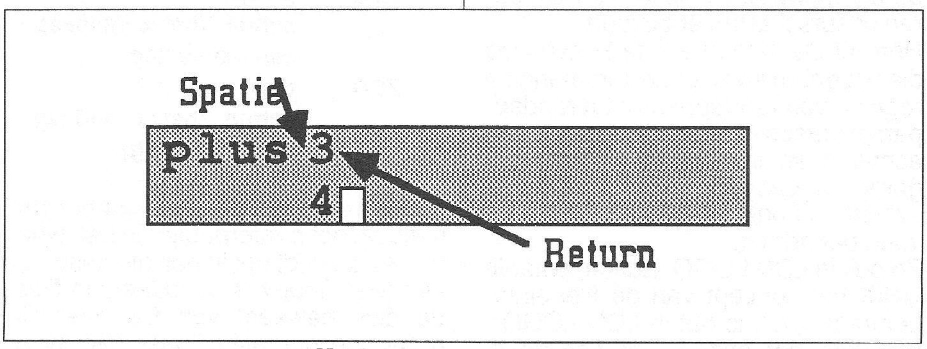


Sub-vlakken

In LCN LOGO kan zo'n vlak worden geëvalueerd ("gerund"). Dit geschiedt door met de cursortoetsen naar het betreffende vlak te gaan (het vlak kleurt nu groen), en de evaluatietoets in te drukken. De inhoud van het vlak wordt dan gevalueerd, en het resultaat wordt afgedrukt op het tekstscherm (is er sprake van grafische uitvoer, dan verschijnt die uiteraard op het grafisch scherm).

Werken met vlakken

Programmeren in LCN LOGO komt neer op het werken met vlakken. Daarom bestaat er een hele reeks van editorkommando's waarmee de gebruiker vlakken kan weghalen, invoegen, kopiëren en verplaatsen. Deze LOGO-editor is dus zeker geen gewone BASIC-achtige teksteditor; het is een krachtig instrument dat de layout en de syntax van LCN LOGO ondersteunt.
Met het oog op het foutvrij maken van programma's ("debuggen") beschikt LCN LOGO over een zg. stepper. Dat is een editor-hulpmiddel waarmee men de volgorde waarin vlakken worden geëvalueerd zichtbaar kan maken: men kan het verloop van een pro-

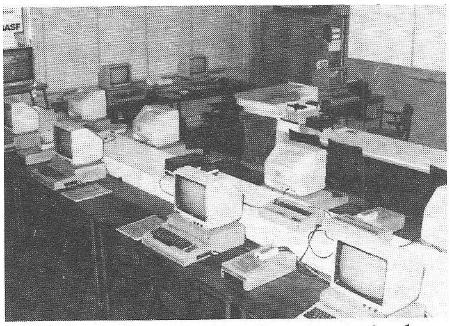


Het intypen van argumenten

gramma volgen doordat de vlakken die door de interpreter worden verwerkt stap voor stap wit kleuren. Daarnaast biedt de editor de mogelijkheid om woorden in de programmatekst op te zoeken en te veranderen. Een dergelijke zoekfaciliteit is, zeker wanneer er sprake is van grotere programma's, onontbeerlijk.

In- en uitvoer met objekten

Onder in- en uitvoer verstaan we alle handelingen die betrekking hebben op randapparaten, zoals de diskdrive, de printer, of de plotter: maar ook het beeldscherm noemen we een randapparaat. Omdat het scherm voor grafische uitvoer en voor tekstuitvoer gebruikt kan worden, spreken we zelfs van twee apparaten. Traditioneel wordt in een LOGO voorzien in een groot aantal primitieve kommando's voor in- en uitvoer van en naar de randapparaten. Zo kennen we laad, bewaar, en inhoud voor de disk: druk,



Steeds meer zien we computers in de leslokalen verschijnen.

typ en drukaf voor het tekstscherm en een systeem van kanalen om met de printer te kunnen werken en, last but not least, alle schildpadkommando's om uitvoer op het grafisch scherm te verkrijgen. Niet zelden wordt deze groep uitgebreid met allerlei gespecialiseerde kommando's, om bijvoorbeeld een scherm te wissen (veeguit, maakschoon) of een diskfile te verwijderen (fileweg), om schermen te kleuren (kleurscherm), enzovoort.
Hoewel de namen en de konventies die het gebruik van deze kommando's regelen van randapparaat tot randapparaat verschillen, zijn de basisideeën achter in- en uitvoer voor elk apparaat gelijk. In geavanceerde besturingssystemen zoals UNIX wordt die eenvoud benadrukt.
Zo ook in LCN LOGO. Echter, waar in UNIX het koncept van de filepointer centraal staat, is dat in LCN LOGO - hoe kan het anders- de schildpad. LOGO is beroemd om zijn schildpad, ontegenzeggelijk een van Paperts meest sterke ideeën. Dit object to think with kombineert begrijpelijkheid met een verrassend groot potentiel aan mogelijke aktiviteiten waarvan menigeen die kinderen met LOGO heeft zien werken onder de indruk is gekomen. De schildpad is in feite de softwarematige representant van de robot, de turtoise (Amerikaans: turtle), die vanuit het toetsenbord bestuurd kon worden. Het blijkt echter dat de schildpad voor meerdere karretjes gespannen kan worden.In LCN LOGO hoort bij elk randapparaat een schildpad-achtig objekt. Zoals de grafische schildpad, kan elk van deze objekten zich verplaatsen. Echter, waar de grafische schildpad zich voortbeweegt over het (grafisch) beeldscherm, beweegt de disk-schildpad (kortweg: disk) zich over een floppy; beweegt de printer-schildpad (de printer) zich over het papier; en beweegt de cursor-schildpad (de cursor) zich over het tekstscherm.
Met al deze schildpadden is het niet op voorhand duidelijk voor wie een opdracht als vooruit 20 bedoeld is. De volgende voorbeelden maken duidelijk hoe deze verwarring voorkomen wordt:

```
ZEG   schildpad
      vooruit 20
ZEG   cursor
      vooruit 20
ZEG   disk
      vooruit 20
```

Elke schildpad (kortweg: elk objekt) wordt aangesproken als we het een opdracht willen laten uitvoeren. De namen van de opdrachten zijn telkens hetzelfde: in kombinatie met een objekt geven ze een ander resultaat. Om de flexibiliteit van dit soort van in- en uitvoer weer te geven, geven we een paar voorbeelden:

```
ZEG   printer
      schrijf "[het is vandaag
      een mooie dag
ZEG   disk
      schrijf "[het is vandaag
      een mooie dag]
```

In het eerste geval verschijnt het zinnetje op het printerpapier: in het tweede geval wordt het in een file geschreven (een floppy is opgedeeld in files: de disk beweegt van file naar file m.b.v. de opdracht: 'naar "file' waar 'file' een filenaam is).

```
ZEG   disk
      naar "mijnteksten
      lees lijst
```

Deze opdracht levert de eerste lijst in de file 'mijnteksten' op. De disk is echter zekerniet het enige objekt dat kan lezen:

```
ZEG   cursor
      lees lijst
```

zorgt ervoor dat de cursor over het tekstscherm beweegt tot hij een lijst tegenkomt. Die lijst wordt gelezen. Deze leesvaardigheid van de cursor maakt het ontwerpen van scherm-menu's bijzonder eenvoudig.

```
ZEG   printer
      schrijf ZEG disk
      lees lijst
```

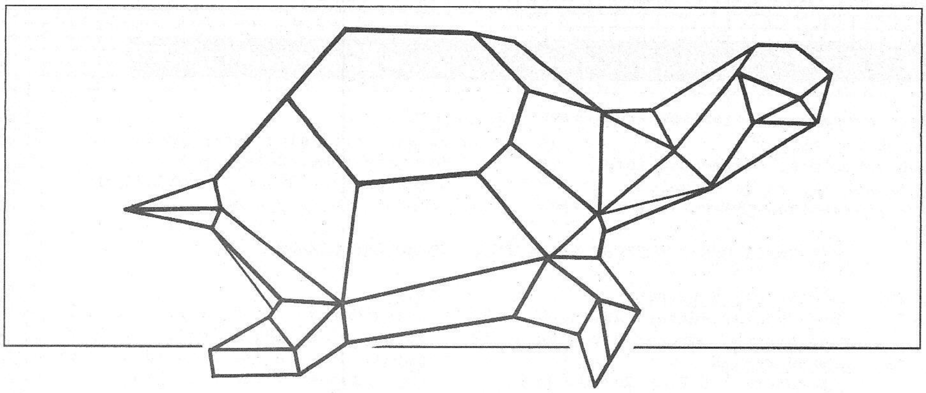
Deze opdracht zorgt voor een 'hard copy' van een deel van een diskfile. De speciale nadruk die gelegd wordt op schildpadden vestigt ook de aandacht op de dingen waarover zij bewegen: het scherm, de floppy, en het printerpapier. Het grafisch scherm en het tekstscherm zijn in LCN LOGO ook objekten.

```
ZEG   gs
      wis
      kleur rood
```

Deze gekombineerde opdracht veegt het grafisch scherm (gs) schoon, en geeft het de kleur rood.
De objekten vormen een zeer interessante benadering van de vele taken van het besturingssysteem van de computer. Ze maken het mogelijk dat de gevorderde leerling experimenteert met zaken als gegevensbestanden op floppy disks, met faciliteiten als hard copy, enzovoort. Bovendien garandeert deze benadering begrijpelijkheid om dezelfde redenen waarom de inmiddels klassieke grafische schildpad begrijpelijk is.

Foutmeldingen

Zoals gezegd wordt de statusbalk gebruikt voor systeem- en foutmeldingen. Die balk is meestal neutraal grijs, maar hij krijgt bij het optreden van fouten een rode waarschuwingskleur. Betreft het een fout in een LOGO-programma, dan wordt bovendien het vlak waarin zich de fout bevindt rood gekleurd.



Is bijvoorbeeld x in het volgende opdracht niet gebonden aan een waarde, dan zal de statusbalk de melding 'x heeft geen waarde' geven, en zal de fout op de volgende manier worden aangewezen: (zie hiervoor figuur 3 op de volgende pagina)

Afkortingen

Iedere LOGO-versie biedt de gebruiker de mogelijkheid om vaak voorkomende opdrachten af te korten: vooruit als 'vt', rechts als 're', enzovoort. Dat bespaart nogal wat typwerk, maar met name langere programma's zijn met al die afkortingen erin moeilijk te lezen. LCN LOGO kent het gebruik van afkortingen ook, maar ze worden -anders dan in andere versies van LOGO- reeds bij het intypen door het systeem herkend. Dientengevolge verschijnen de opdrachten voluit op het scherm! De leerling wil bijvoorbeeld een vierkantje tekenen, en typt:

```
H       4
        vt 50
        rs 90
```

Op het scherm verschijnt het ingetypte als:

```
HERHAAL4
        vooruit 50
        rechts 90
```

Driedimensionaal

Een van de meest interessante kenmerken van LCN LOGO is ongetwijfeld de mogelijkheid om met de grafische schildpad in drie dimensies te werken. Daarvoor bestaan speciale kommando's, zoals bijvoorbeeld kantel en duik.
Tekenen in de derde dimensie vergt enige oefening (de schildpad zelf is dan onzichtbaar), maar na verloop van tijd zijn de resultaten al gauw spektakulair.

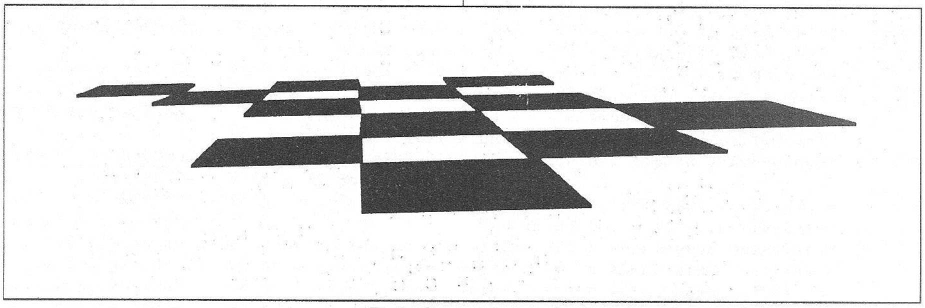
Windows

Van een serieus softwarepakket mag men tegenwoordig verwachten dat de bediening in belangrijke mate menugestuurd is. Bij LCN LOGO is dat dan ook het geval. Maar er is meer: met behulp van eenvoudige primitieven kan men kleine tekst-schermpjes maken. Deze schermpjes scrollen onafhankelijk van elkaar, en de gebruiker kan ze een naam en kleur geven. Het is zelfs mogelijk zo'n tekst-raampje over het scherm te laten wandelen! Daarbij blijft de achtergrond (het tekstscherm) gehandhaafd.

Zoals gezegd heeft LCN LOGO een 'lezende' cursor: zet je de cursor op een woord, dan kun je hem dat woord laten opleveren. Tekst-windows en cursor-repertoire vormen tesamen de ingrediënten voor zelfgemaakte menu's. Dat men hier fantastische resultaten mee kan bereiken is te zien aan het meegeleverde demonstratie-programma.

Opties

Zoals al eerder gesteld is de Commodoreversie ook nog voorzien van sproken: schildpadklonen, zodat met meerdere objecten tegelijk gewerkt kan worden. Ook biedt de Commodoreversie uitgebreide geluidsmogelijkheden. In het kader van deze bespreking weiden we daar voorlopig niet verder over uit. Maar dat de Commodore met deze LCN-LOGO nu heel bijzondere dingen kan moet duidelijk zijn. Niet voor niets maakt de LCN-LOGO al een tijdje deel uit van het LOI onderwijspakket voor de Commodore. Ook het NIVOproject zal LCN-LOGO binnenkort omarmen als één van de beste softwarepakketten voor het middelbaar onderwijs.
Daar werkt men met MS-DOSapparatuur. Niet alle MS-DOS computers zijn hetzelfde. Daar zal een ieder het inmiddels over eens zijn. Vandaar de noodzaak om programma's zo flexibel te maken, dat ze ook op andere konfiguraties draaien.
Bij LCN LOGO is gekozen voor een optiefile. Dat is een klein bestand met de naam LOGO.OPT, waarin de default-waarden van een aantal essentiële variabelen opgenomen zijn. Men kan die waarden met iedere eenvoudige tekstverwerker naar eigen behoefte veranderen. Het LOGO.OPT file bevat bijvoorbeeld waarden voor de afmetingen van de opslagruimte (uitgedrukt in aantal cellen) en voor de rekursiediepte (uitgedrukt in niveaus). Bovendien biedt LOGO.OPT de mogelijkheid naar zwart-wit te switchen.



Driedimensionaal

De prijs van de LCN-LOGO incl. BTW en verzendkosten bedraagt *f* 229,- (Commodore) en *f* 259,- voor de MS-DOS versie.
Voor degenen die eens kennis willen maken met de mogelijkheden van het LCD-LOGO voor MS-DOS machines-bestaat er een demo-versie. Helaas kunnen de commodore gebruikers niet van een dergelijk aanbod gebruik maken; een demo-versie voor de commodore is helaas niet voorhanden.
Voor verdere inhoudelijke informatie over LCN-LOGO kan men bellen naar: 080-238130

Print-out Met o.a. Adresboek, Gokken, Flash, Orgel, Kubus

Dit is de eerste PRINT-OUT rubriek van het nieuwe jaar. Ik wil van de gelegenheid gebruik maken om u namens alle medewerkers van de uitgeverij Sala Communications een heel gelukkig en met de computer een heel produktief 1988 toe te wensen. We krijgen voor deze rubriek nog steeds veel programma's toegestuurd. Het valt echter op dat hierbij bijna nooit programma's van vrouwen zijn. Dat is iets wat in dit nieuwe jaar wel mag veranderen. Wat de dames wel veel mogen doen is het overtikken van de listings, zoals bij de listingtelefoon duidelijk is te merken aan de reacties. Denkt u er nog aan dat programma's die mee willen dingen naar één van de mooie prijzen voor de prijsvraag bij ons binnen moeten zijn op 15 februari van dit jaar.
R. Goudriaan.

Syntax Checksum

Het overtikken van een listing kan een heel karwei zijn en als u een beetje normaal mens bent dan maakt u daarin beslist een aantal fouten. Nu is niets moeilijker om de fouten uit je eigen werk te halen. Al geruime tijd geleden heeft Jan Bodzinga hiervoor een zgn. Checksum-programma geschreven. Om de vele nieuwe lezers van Commodore-info te helpen volgt hieronder nog een keer een volledige uitleg over de werking van dit programma, waarmee het, hoe vreemd dat misschien ook lijkt, echt mogelijk is om de fouten in elke door ons geplaatste listing op te sporen.

Hiervoor gaat u als volgt te werk:
1. U tikt de listing heel zorgvuldig over en SAVEt hem voordat u het programma RUNt op een diskette of cassette.
2. U tikt het RUN commando in. Mocht het programma de boodschap 'FOUT in dataregels!' geven dan heeft u een fout bij het overtikken gemaakt. Herstel dan de fout en SAVE de verbeterde versie. Mocht het programma met de boodschap 'data is weggezet checksum testen met sys...' komen dan is tot dusver alles goed. Het programma is nu in een stukje machinetaalgeheugen gezet. Als u het NEW commando geeft blijft het toch in de computer staan.
Alle door ons geplaatste programma's zijn in Basic geschreven.
Als u een programma heeft overgetikt SAVE het eerst, mocht er iets mis gaan dan hoeft u niet de gehele listing opnieuw te gaan intikken. Als u nu een programma op fouten wilt gaan controleren dan kunt u dat in het geheugen laden(wel eerst het checksum programma hebben gerund). Vervolgens typt u zonder het programma te runnen de opdracht sys 49152(c-64) of sys 1536 (c-16 en plus/4)in.
Als alles goed is gegaan loopt er nu een rij regelnummers over het scherm met getallen erachter. Dezelfde lijst staat ook achter elk door ons geplaatste programma. Wijkt nu een nummer achter een regelnummer af van het nummer dat in het blad staat dan heeft u in die regel iets anders ingetikt dan er in het blad stond. U kunt de stroom getallen d.m.v. de RUN/STOP toets pauzeren en weer vervolgen met de F1 of F7 toets. Het is uitermate belangrijk dat u goed met dit programma overweg kunt en mocht u het niet goed werkend krijgen bel dan gerust even met onze listingservice telefoonlijn. (Maandag 17.00 - 21.00 uur. Telefoonnummer 02155-25162.)

```
1     rem ******************************
      ***
2     rem basic loader "SYNTAX.CHECKSUM"
3     rem na de commando's "run" en "new
      "
4     rem blijft dit programma in het ge
      -
5     rem heugen. laad het te testen pro
      -
6     rem gramma en tik daarna sys 49152
      .
7     rem ******************************
      ***
10    i=49152   :rem beginadres
20    reada:ifa<0then40:rem data ingelez
      en
30    pokei,a:i=i+1:b=b+a:goto20
40    if b<>16844thenprint"[SHIFT-CLR]fo
      ut[SPACE]in[SPACE]dataregels!":b=0
      :end
50    poke49184,148:poke49185,192
55    i=49300
60    read a: ifa<0then80
70    pokei,a:b=a+b:i=i+1:goto60
80    if b<>20068thenprint"[SHIFT-CLR]fo
      ut[SPACE]in[SPACE]dataregels![SPAC
      E](vanaf[SPACE]regel[SPACE]240)":b
      =0:end
90    print"data[SPACE]is[SPACE]weggezet
      "
95    print"checksum[SPACE]testen[SPACE]
      met[SPACE]sys49152"
100   data 165,43,166,44,133,163,134,164
      ,169,147
110   data 32,210,255,160,0,240,3,32,73,
      192
120   data 32,73,192,208,1,96,32,225,255
      ,208
130   data 3,76,116,164,32,81,192,32,73,
      192
140   data 240,12,201,32,240,247,24,101,
      167,133
150   data 167,76,37,192,166,167,169,0,1
      32,168
160   data 32,205,189,169,13,32,210,255,
      164,168
170   data 76,17,192,200,208,2,230,164,1
      77,163
180   data 96,162,0,189,123,192,240,6,32
      ,210
190   data 255,232,208,245,32,73,192,170
      ,32,73
200   data 192,132,168,32,205,189,162,3,
      169,32
```

```
210     data 32,210,255,202,208,250,169,0,
        133,167
220     data 164,168,96,82,69,71,69,76,32,
        0
230     data  -1
240     data 165,197,201,3,240,7,201,4,240
250     data 6,76,148,192,76,34,192,169
260     data 147,32,210,255,76,161,192
270     data -1
```

** EINDE LISTING checksum 64 **

REGEL	1	249	REGEL	100	183
REGEL	2	84	REGEL	110	158
REGEL	3	105	REGEL	120	232
REGEL	4	2	REGEL	130	183
REGEL	5	246	REGEL	140	96
REGEL	6	152	REGEL	150	96
REGEL	7	249	REGEL	160	127
REGEL	10	157	REGEL	170	71
REGEL	20	64	REGEL	180	223
REGEL	30	38	REGEL	190	73
REGEL	40	57	REGEL	200	79
REGEL	50	14	REGEL	210	109
REGEL	55	251	REGEL	220	106
REGEL	60	192	REGEL	230	225
REGEL	70	42	REGEL	240	16
REGEL	80	244	REGEL	250	163
REGEL	90	245	REGEL	260	92
REGEL	95	237	REGEL	270	22

Adresboek 64

Dat in Belgie ons blad ook heel veel gelezen wordt mag wel blijken uit de regelmatig ingezonden programma's. Zo ook deze inzending uit Belgie, die is van W van Thurenhout uit Katelijne Waver. Adresboek kan natuurlijk door iedereen die een beetje kan programmeren gewijzigd worden en er kunnen al naar gelang de behoefte uitbreidingen in aangebracht worden. Hierdoor is het mogelijk dat er in plaats van adressen een postzegel- of platen-bestand van wordt gemaakt. In elk veld kan gezocht worden naar een gegeven uit het bestand. Het programma is volledig menugestuurd.

```
10      printchr$(147)
20      printchr$(14)
30      printtab(15)"[9xCRSR-DOWN][CTRL-2]
        Programma":print
40      printtab(10)"[CTRL-8]Willy[SPACE]V
        an[SPACE]Thurenhout[COM-7]":fori=1
        to2000:next
50      poke 53280,5:poke 53281,0:printchr
        $(154);:dimd$(200,6)
60      gosub 2000
70      print"kies[SPACE]de[SPACE]gewenste
        [SPACE]functie[SPACE]:"
80      print"--------------------------":
        print
90      print"[7xSPACE]-1-[SPACE]bestand[S
        PACE]laden"
100     print"[7xSPACE]-2-[SPACE]bestand[S
        PACE]saven"
110     print"[7xSPACE]-3-[SPACE]gegevens[
        SPACE]invoeren"
120     print"[7xSPACE]-4-[SPACE]gegevens[
        SPACE]veranderen"
130     print"[7xSPACE]-5-[SPACE]gegevens[
        SPACE]selecteren/uitvoeren";
140     print"[7xSPACE]-6-[SPACE]gegevens[
        SPACE]wissen"
150     print"[7xSPACE]-0-[SPACE]programma
        [SPACE]beeindigen"
160     print
170     print"[SPACE]keuze[SPACE](0-6):[SP
        ACE]"
180     getx$:ifx$<"0"orx$>"6"then180
190     ifx$<>"0"then 250
200     print:print"[12xSPACE]zeker[SPACE]
        (j/n)?"
210     getx$:ifx$<>"n"andx$<>"j"then210
220     if x$="n"then60
230     gosub2000
240     end
250     onval(x$)gosub270,450,590,870,1160
        ,1760
260     goto60
270     rem **************
280     rem gegevens laden
290     rem **************
300     gosub 2000
310     input"naam[SPACE]van[SPACE]het[SPA
        CE]bestand[SPACE]:";dn$
320     open15,8,15
330     open1,8,2,dn$+",s,r"
340     input#15,fe:if fe=0then370
350     print"diskettefout!!!"
360     goto420
370     x=1
380     input#1,d$(x,1),d$(x,2),d$(x,3),d$
        (x,4),d$(x,5),d$(x,6)
390     if st<>64 then x=x+1:goto380
400     print"bestand[SPACE]is[SPACE]gelad
        en[SPACE]en[SPACE]bevat[SPACE]";x
410     print"gegevensrecords":print
420     close1:close15
430     print"verder[SPACE]met[SPACE]retur
        n"
440     inputx$:return
450     rem **************
460     rem gegevens saven
470     rem ****************
480     if x>0then 500
490     gosub2180:return
500     gosub 2000
510     open1,8,2,"`:"+dn$+",s,w"
520     fori=1tox
530     print#1,d$(i,1)","d$(i,2)","d$(i,3
        )",";
540     print#1,d$(i,4)","d$(i,5)","d$(i,6
        )
550     next
560     print"gegevens[SPACE]zijn[SPACE]ge
        saved":close 1:print
570     print"verder[SPACE]met[SPACE]retur
        n"
580     inputx$:return
590     rem *****************
600     rem gegevens invoeren
610     rem *****************
620     ifx>0then goto640
630     gosub2000:input"bestandnaam[SPACE]
        ";dn$
640     x=x+1
650     gosub2000
```

```
660   print"gegevensinvoer:"
670   print"---------------":print
680   print"slechts[SPACE]25[SPACE]aansl
      agen[SPACE]a.u.b."
690   print"---------------------------"
      :print
700   i=x:gosub2080
710   fori=1to6:printchr$(145);:next
720   fori=1to6:printtab(11);:inputd$(x,
      i):nexti
730   if len(d$(x,1))>25 then 2620
740   if len(d$(x,2))>25 then 2730
750   if len(d$(x,3))>25 then 2840
760   if len(d$(x,4))>25 then 2940
770   if len(d$(x,5))>25 then 3050
780   if len(d$(x,6))>25 then 3160
790   print:print"correct[SPACE](j/n)?"
800   getx$:ifx$<>"n"andx$<>"j"then800
810   ifx$="j"then830
820   goto650
830   print"nog[SPACE]meer[SPACE]invoere
      n[SPACE](j/n)?"
840   getx$:ifx$<>"j"andx$<>"n"then840
850   ifx$="j"then640
860   return
870   rem ******************
880   rem gegevens veranderen
890   rem ******************
900   ifx>0then920
910   gosub2180:return
920   gosub2000
930   input"naam[2xSPACE]:[SPACE]";n1$
940   fori=1tox
950   ifd$(i,1)=n1$then990
960   next i
970   print"naam[SPACE]niet[SPACE]gevond
      en!"
980   print"verder[SPACE]met[SPACE]retur
      n":inputx$:return
990   gosub2000
1000  print"-1-[SPACE]naam[7xSPACE]:";d$
      (i,1)
1010  print"-2-[SPACE]straat/nr.[SPACE]:
      ";d$(i,2)
1020  print"-3-[SPACE]postcode[3xSPACE]:
      ";d$(i,3)
1030  print"-4-[SPACE]plaats[5xSPACE]:";
      d$(i,4)
1040  print"-5-[SPACE]telefoon[3xSPACE]:
      ";d$(i,5)
1050  print"-6-[SPACE]relatie[4xSPACE]:"
      ;d$(i,6):print:print
1060  print"nr.[SPACE]van[SPACE]het[SPAC
      E]te[SPACE]veranderen[SPACE]veld:"
      :print"7=geen[SPACE]verandering":p
      rint
1070  getx$:ifval(x$)<1 or val(x$)>6andv
      al(x$)<>7then1070
1080  if val(x$)=7then1120
1090  y=val(x$)
1100  input"nieuwe[SPACE]inhoud";d$(i,y)
      :print
1110  goto 990
1120  print"meer[SPACE]veranderen[SPACE]
      (j/n)?"
1130  getx$:ifx$<>"j"andx$<>"n"then1130
1140  ifx$="j"then870
1150  return
1160  rem *****************************
1170  rem gegevens selecteren/uitvoeren
1180  rem *****************************
1190  ifx>0then1210
1200  gosub2180:return
1210  goto 2250
1220  ifx>0thengosub2000:print"uitvoer[S
      PACE]printer[SPACE]of[SPACE]beelds
      cherm[SPACE]?":print
1230  print"[SPACE]p[SPACE]=[SPACE]print
      er":print:print"[SPACE]b[SPACE]=[S
      PACE]beeldscherm"
1240  getx$:ifx$<>"p"andx$<>"b"then1240
1250  o$=x$:ifo$="b"then1300
1260  print:print"papier[SPACE]of[SPACE]
      etiketten[SPACE]?":print
1270  print"[SPACE]p[SPACE]=[SPACE]lijst
      [SPACE]op[SPACE]papier":print:prin
      t"[SPACE]e[SPACE]=[SPACE]adressenl
      abels"
1280  getx$:ifx$<>"p"andx$<>"e"then1280
1290  d$=x$
1300  gosub2000
1310  print"geef[SPACE]de[SPACE]trefwoor
      den[SPACE]aan[SPACE]:"
1320  print"bij[SPACE]niet[SPACE]relevan
      te[SPACE]velden[SPACE]alleen[SPACE
      ]return!";
1330  print"----------------------------
      ------------":print
1340  i=0:gosub2080
1350  fori=1to6:printchr$(145);:s$(i)=""
      :next
1360  fori=1to6:printtab(12);:inputs$(i)
      :next
1370  ifo$="b" or d$="e"then1470
1380  gosub2000:print"printer[SPACE]aang
      eschakeld[SPACE](j)?"
1390  getx$:ifx$<>"j"then1390
1400  open1,4
1410  a$="naam":b$="adres":c$="code":g$=
      "woonplaats":e$="telefoon":f$="rel
      atie"
1420  print#1,a$;spc(26-len(a$))b$;spc(2
      6-len(b$))c$;
1430  print#1,spc(4)g$
1440  print#1,e$;spc(15-len(e$))f$
1450  fori=1to79:print#1,"=";:next:print
      #1
1460  close1
1470  fori=1tox
1480  fory=1to6
1490  ifs$(y)=left$(d$(i,y),len(s$(y)))t
      henz=z+1:goto1500
1500  nexty
1510  ifz=6then gosub1570
1520  z=0:nexti
1530  print:print"einde[SPACE]van[SPACE]
      het[SPACE]bestand[SPACE]!!":print
1540  print"verder[SPACE]met[SPACE]retur
      n":print
1550  inputx$
1560  return
1570  ifo$="b"then1720
1580  ifd$="e"then1650
1590  open1,4,7
1600  print#1,d$(i,1);spc(26-len(d$(i,1)
      ))d$(i,2);spc(26-len(d$(i,2)))d$(i
      ,3);
1610  print#1,spc(4)d$(i,4)
1620  print#1,d$(i,5);spc(15-len(d$(i,5)
      ))d$(i,6)
```

```
1630 print#1:close1
1640 return
1650 open2,4,7
1660 print#2
1670 print#2,d$(i,1)
1680 print#2,d$(i,2)
1690 print#2,d$(i,3)"[3xSPACE]"d$(i,4);
     chr$(13);chr$(13);chr$(13);chr$(13
     );chr$(13)
1700 close2
1710 return
1720 gosub2000:gosub2080
1730 print:print"verder[SPACE](j)?"
1740 getx$:ifx$<>"j"then1740
1750 return
1760 rem ***************
1770 rem gegevens wissen
1780 rem ***************
1790 ifx>0then1810
1800 gosub2180:return
1810 gosub2000
1820 input"naam[2xSPACE]:[SPACE]";n1$
1830 fori=1tox
1840 ifd$(i,1)=n1$then1880
1850 nexti
1860 print"naam[SPACE]niet[SPACE]gevond
     en!":print
1870 print"verder[SPACE]met[SPACE]retur
     n":inputx$:return
1880 gosub2000:gosub2080
1890 print:print"adres[SPACE]wissen[SPA
     CE](j/n)?"
1900 getx$:ifx$<>"j"andx$<>"n"then1900
1910 ifx$="n"thenreturn
1920 fory=itox-1
1930 forj=1to6
1940 d$(y,j)=d$(y+1,j)
1950 nextj,y
1960 forj=1to6:d$(x,j)="":nextj
1970 x=x-1
1980 print"record[SPACE]is[SPACE]gewist
     !"
1990 inputx$:return
2000 rem ***************
2010 rem programma titel
2020 rem ***************
2030 printchr$(147);
2040 printtab(6);"=====================
     ========"
2050 printtab(6);"A[SPACE]D[SPACE]R[SPA
     CE]E[SPACE]S[SPACE]S[SPACE]E[SPACE
     ]N[SPACE]B[SPACE]E[SPACE]S[SPACE]T
     [SPACE]A[SPACE]N[SPACE]D"
2060 printtab(6);"=====================
     ========";:print"[COM-1]";x"[COM-7
     ]":print:print
2070 return
2080 rem *************
2090 rem recorduitvoer
2100 rem *************
2110 print"naam[5xSPACE]:";"[3xSPACE]";
     d$(i,1)
2120 print"straat/nr:";"[3xSPACE]";d$(i
     ,2)
2130 print"postcode[SPACE]:";"[3xSPACE]
     ";d$(i,3)
2140 print"plaats[3xSPACE]:";"[3xSPACE]
     ";d$(i,4)
2150 print"telefoon[SPACE]:";"[3xSPACE]
     ";d$(i,5)
2160 print"relatie[2xSPACE]:";"[3xSPACE
     ]";d$(i,6)
2170 return
2180 rem **************
2190 rem geen bestand !
2200 rem **************
2210 gosub2000
2220 print"geen[SPACE]bestand[SPACE]in[
     SPACE]de[SPACE]computer!":print
2230 print"verder[SPACE]met[SPACE]retur
     n"
2240 inputx$:return
2250 gosub2000:print"volledige[SPACE]ui
     tvoer[SPACE]of[SPACE]enkel[SPACE]a
     dres[SPACE]?":print
2260 print"[SPACE]v[SPACE]=[SPACE]lijst
     [SPACE]met[SPACE]alle[SPACE]gegeve
     ns":print:print"[SPACE]a[SPACE]=[S
     PACE]enkel[SPACE]adressenlijst"
2270 getx$:ifx$<>"v" and x$<>"a" then 2
     270
2280 v$=x$:ifv$="v"then 1220
2290 gosub 2000:print"geef[SPACE]de[SPA
     CE]trefwoorden[SPACE]aan[SPACE]:"
2300 print"bij[SPACE]niet[SPACE]relevan
     te[SPACE]velden[SPACE]alleen[SPACE
     ]return!";
2310 print"----------------------------
     ------------":print
2320 i=0:gosub2570
2330 fori=1to4:printchr$(145);:s$(i)=""
     :next
2340 fori=1to4:printtab(12);:inputs$(i)
     :next
2350 gosub2000:print"printer[SPACE]aang
     eschakeld[SPACE](j)?"
2360 getx$:ifx$<>"j"then2360
2370 open3,4,7
2380 a$="naam":b$="adres":c$="code":g$=
     "woonplaats"
2390 print#3,a$;spc(26-len(a$))b$;spc(2
     6-len(b$))c$;spc(4)g$
2400 for i=1 to79:print#3,"=";:next:pri
     nt#3
2410 close3
2420 fori=1tox
2430 fory=1to4
2440 ifs$(y)=left$(d$(i,y),len(s$(y)))t
     henz=z+1:goto2450
2450 nexty
2460 ifz=4thengosub2520
2470 z=0:nexti
2480 print:print"einde[SPACE]van[SPACE]
     het[SPACE]bestand[SPACE]!!":print
2490 print"verder[SPACE]met[SPACE]retur
     n":print
2500 inputx4
2510 return
2520 open1,4,7
2530 print#1,d$(i,1);spc(26-len(d$(i,1)
     ))d$(i,2);spc(26-len(d$(i,2)))d$(i
     ,3);
2540 print#1,spc(4)d$(i,4)
2550 print#1:close1
2560 return
2570 print"naam[5xSPACE]:";d$(i,1)
2580 print"straat/nr:";d$(i,2)
2590 print"postcode[SPACE]:";d$(i,3)
2600 print"plaats[3xSPACE]:";d$(i,4)
2610 return
```

```
2620  print chr$(147)
2630  print"[2xCRSR-DOWN]"
2640  printspc((40-len(d$(x,1)))/2)"[CTR
      L 9]";d$(x,1)
2650  printtab(14)"[2xCRSR-DOWN][CTRL-9]
      is[SPACE]te[SPACE]lang.[CTRL-0]"
2660  print"[3xCRSR-DOWN]"
2670  printtab(8)"[CTRL-5]Druk[SPACE]toe
      ts[SPACE]voor[SPACE]vervolg[COM-7]
      "
2680  getk$:ifk$=""then2680
2690  fory=1to6:d$(x,y)="":nexty
2700  x=x-1:print"[2xCRSR-DOWN][7xCRSR-R
      IGHT]Record[SPACE]is[SPACE]volledi
      g[SPACE]gewist!":print:print:print
2710  printtab(13)"[CTRL-8]Druk[SPACE]op
      [SPACE]<return>[COM-7]"
2720  inputx$:return
2730  print chr$(147)
2740  print"[2xCRSR-DOWN]"
2750  printspc((40-len(d$(x,2)))/2)"[CTR
      L 9]";d$(x,2)
2760  printtab(14)"[2xCRSR-DOWN][CTRL-9]
      is[SPACE]te[SPACE]lang.[CTRL-0]"
2770  print"[3xCRSR-DOWN]"
2780  printtab(8)"[CTRL-5]Druk[SPACE]toe
      ts[SPACE]voor[SPACE]vervolg[COM-7]
      "
2790  getk$:ifk$=""then2850
2800  fory=1to6:d$(x,y)="":nexty
2810  x=x-1:print"[2xCRSR-DOWN][7xCRSR-R
      IGHT]Record[SPACE]is[SPACE]volledi
      g[SPACE]gewist!":print:print:print
2820  printtab(13)"[CTRL-8]Druk[SPACE]op
      [SPACE]<return>[COM-7]"
2830  inputx$:return
2840  print chr$(147)
2850  print"[2xCRSR-DOWN]"
2860  printspc((40-len(d$(x,3)))/2)"[CTR
      L 9]";d$(x,3)
2870  printtab(14)"[2xCRSR-DOWN][CTRL-9]
      is[SPACE]te[SPACE]lang.[CTRL-0]"
2880  print"[3xCRSR-DOWN]"
2890  printtab(8)"[CTRL-5]Druk[SPACE]toe
      ts[SPACE]voor[SPACE]vervolg[COM-7]
      "
2900  getk$:ifk$=""then2900
2910  fory=1to6:d$(x,y)="":nexty
2920  x=x-1:print"[2xCRSR-DOWN][7xCRSR-R
      IGHT]Record[SPACE]is[SPACE]volledi
      g[SPACE]gewist!":print:print:print
2930  printtab(13)"[CTRL-8]Druk[SPACE]op
      [SPACE]<return>[COM-7]"
2940  print chr$(147)
2950  print"[3xCRSR-DOWN]"
2960  printspc((40-len(d$(x,4)))/2)"[CTR
      L 9]";d$(x,4)
2970  printtab(14)"[2xCRSR-DOWN][CTRL-9]
      is[SPACE]te[SPACE]lang.[CTRL-0]"
2980  print"[3xCRSR-DOWN]"
2990  printtab(8)"[CTRL-5]Druk[SPACE]toe
      ts[SPACE]voor[SPACE]vervolg[COM-7]
      "
3000  getk$:ifk$=""then3000
3010  fory=1to6:d$(x,y)="":nexty
3020  x=x-1:print"[2xCRSR-DOWN][7xCRSR-R
      IGHT]Record[SPACE]is[SPACE]volledi
      g[SPACE]gewist!":print:print:print
3030  printtab(13)"[CTRL-8]Druk[SPACE]op
      [SPACE]<return>[COM-7]"
3040  inputx$:return
3050  print chr$(147)
3060  print"[3xCRSR-DOWN]"
3070  printspc((40-len(d$(x,5)))/2)"[CTR
      L 9]";d$(x,5)
3080  printtab(14)"[2xCRSR-DOWN][CTRL-9]
      is[SPACE]te[SPACE]lang.[CTRL-0]"
3090  print"[3xCRSR-DOWN]"
3100  printtab(8)"[CTRL-5]Druk[SPACE]toe
      ts[SPACE]voor[SPACE]vervolg[COM-7]
      "
3110  getk$:ifk$=""then3110
3120  fory=1to6:d$(x,y)="":nexty
3130  x=x-1:print"[2xCRSR-DOWN][7xCRSR-R
      IGHT]Record[SPACE]volledig[SPACE]i
      s[SPACE]gewist!":print:print:print
3140  printtab(13)"[CTRL-8]Druk[SPACE]op
      [SPACE]<return>[COM-7]"
3150  inputx$:return
3160  print chr$(147)
3170  print"[2xCRSR-DOWN]"
3180  printspc((40-len(d$(x,6)))/2)"[CTR
      L 9]";d$(x,6)
3190  printtab(14)"[2xCRSR-DOWN][CTRL-9]
      is[SPACE]te[SPACE]lang.[CTRL-0]"
3200  print"[3xCRSR-DOWN]"
3210  printtab(8)"[CTRL-5]Druk[SPACE]toe
      ts[SPACE]voor[SPACE]vervolg[COM-7]
      "
3220  getk$:ifk$=""then3220
3230  fory=1to6:d$(x,y)="":nexty
3240  x=x-1:print"[2xCRSR-DOWN][7xCRSR-R
      IGHT]Record[SPACE]is[SPACE]volledi
      g[SPACE]gewist!":print:print:print
3250  printtab(13)"[CTRL-8]Druk[SPACE]op
      [SPACE]<return>[COM-7]"
3260  inputx$:return
```

** EINDE LISTING adresboek **

REGEL	10	77	REGEL	170	127	REGEL	330	95	REGEL	490	32	REGEL	650	79
REGEL	20	22	REGEL	180	28	REGEL	340	20	REGEL	500	79	REGEL	660	62
REGEL	30	166	REGEL	190	29	REGEL	350	219	REGEL	510	204	REGEL	670	83
REGEL	40	87	REGEL	200	136	REGEL	360	31	REGEL	520	169	REGEL	680	70
REGEL	50	87	REGEL	210	171	REGEL	370	59	REGEL	530	160	REGEL	690	111
REGEL	60	79	REGEL	220	88	REGEL	380	96	REGEL	540	254	REGEL	700	228
REGEL	70	60	REGEL	230	79	REGEL	390	66	REGEL	550	130	REGEL	710	3
REGEL	80	66	REGEL	240	128	REGEL	400	2	REGEL	560	73	REGEL	720	13
REGEL	90	205	REGEL	250	145	REGEL	410	22	REGEL	570	107	REGEL	730	150
REGEL	100	231	REGEL	260	239	REGEL	420	17	REGEL	580	201	REGEL	740	153
REGEL	110	36	REGEL	270	219	REGEL	430	107	REGEL	590	89	REGEL	750	156
REGEL	120	169	REGEL	280	71	REGEL	440	201	REGEL	600	73	REGEL	760	158
REGEL	130	213	REGEL	290	219	REGEL	450	219	REGEL	610	89	REGEL	770	152
REGEL	140	154	REGEL	300	79	REGEL	460	96	REGEL	620	142	REGEL	780	155
REGEL	150	215	REGEL	310	216	REGEL	470	219	REGEL	630	97	REGEL	790	25
REGEL	160	153	REGEL	320	251	REGEL	480	0	REGEL	640	61	REGEL	800	176

REGEL 810	137	REGEL 1310	214	REGEL 1810	79	REGEL 2310	184	REGEL 2810	224
REGEL 820	36	REGEL 1320	51	REGEL 1820	254	REGEL 2320	192	REGEL 2820	244
REGEL 830	167	REGEL 1330	184	REGEL 1830	169	REGEL 2330	66	REGEL 2830	201
REGEL 840	180	REGEL 1340	188	REGEL 1840	183	REGEL 2340	78	REGEL 2840	77
REGEL 850	136	REGEL 1350	68	REGEL 1850	203	REGEL 2350	255	REGEL 2850	255
REGEL 860	142	REGEL 1360	80	REGEL 1860	116	REGEL 2360	194	REGEL 2860	44
REGEL 870	173	REGEL 1370	252	REGEL 1870	110	REGEL 2370	149	REGEL 2870	89
REGEL 880	205	REGEL 1380	255	REGEL 1880	224	REGEL 2380	210	REGEL 2880	16
REGEL 890	173	REGEL 1390	196	REGEL 1890	79	REGEL 2390	255	REGEL 2890	199
REGEL 900	6	REGEL 1400	48	REGEL 1900	226	REGEL 2400	111	REGEL 2900	172
REGEL 910	32	REGEL 1410	103	REGEL 1910	128	REGEL 2410	211	REGEL 2910	114
REGEL 920	79	REGEL 1420	143	REGEL 1920	173	REGEL 2420	169	REGEL 2920	224
REGEL 930	254	REGEL 1430	99	REGEL 1930	136	REGEL 2430	149	REGEL 2930	244
REGEL 940	169	REGEL 1440	96	REGEL 1940	157	REGEL 2440	213	REGEL 2940	77
REGEL 950	136	REGEL 1450	107	REGEL 1950	81	REGEL 2450	219	REGEL 2950	16
REGEL 960	203	REGEL 1460	209	REGEL 1960	69	REGEL 2460	200	REGEL 2960	46
REGEL 970	161	REGEL 1470	169	REGEL 1970	62	REGEL 2470	65	REGEL 2970	89
REGEL 980	110	REGEL 1480	151	REGEL 1980	44	REGEL 2480	241	REGEL 2980	16
REGEL 990	79	REGEL 1490	208	REGEL 1990	201	REGEL 2490	62	REGEL 2990	199
REGEL 1000	89	REGEL 1500	219	REGEL 2000	5	REGEL 2500	17	REGEL 3000	164
REGEL 1010	10	REGEL 1510	206	REGEL 2010	183	REGEL 2510	142	REGEL 3010	114
REGEL 1020	161	REGEL 1520	65	REGEL 2020	5	REGEL 2520	147	REGEL 3020	224
REGEL 1030	7	REGEL 1530	241	REGEL 2030	136	REGEL 2530	113	REGEL 3030	244
REGEL 1040	160	REGEL 1540	62	REGEL 2040	3	REGEL 2540	90	REGEL 3040	201
REGEL 1050	242	REGEL 1550	1	REGEL 2050	240	REGEL 2550	212	REGEL 3050	77
REGEL 1060	11	REGEL 1560	142	REGEL 2060	237	REGEL 2560	142	REGEL 3060	16
REGEL 1070	204	REGEL 1570	167	REGEL 2070	142	REGEL 2570	206	REGEL 3070	48
REGEL 1080	113	REGEL 1580	161	REGEL 2080	177	REGEL 2580	80	REGEL 3080	89
REGEL 1090	157	REGEL 1590	147	REGEL 2090	124	REGEL 2590	20	REGEL 3090	16
REGEL 1100	242	REGEL 1600	113	REGEL 2100	177	REGEL 2600	121	REGEL 3100	199
REGEL 1110	43	REGEL 1610	90	REGEL 2110	77	REGEL 2610	142	REGEL 3110	166
REGEL 1120	71	REGEL 1620	78	REGEL 2120	207	REGEL 2620	77	REGEL 3120	114
REGEL 1130	221	REGEL 1630	212	REGEL 2130	147	REGEL 2630	255	REGEL 3130	224
REGEL 1140	141	REGEL 1640	142	REGEL 2140	248	REGEL 2640	40	REGEL 3140	244
REGEL 1150	142	REGEL 1650	148	REGEL 2150	144	REGEL 2650	89	REGEL 3150	201
REGEL 1160	81	REGEL 1660	202	REGEL 2160	59	REGEL 2660	16	REGEL 3160	77
REGEL 1170	189	REGEL 1670	85	REGEL 2170	142	REGEL 2670	199	REGEL 3170	255
REGEL 1180	81	REGEL 1680	86	REGEL 2180	219	REGEL 2680	177	REGEL 3180	50
REGEL 1190	47	REGEL 1690	144	REGEL 2190	208	REGEL 2690	114	REGEL 3190	89
REGEL 1200	32	REGEL 1700	210	REGEL 2200	219	REGEL 2700	224	REGEL 3200	16
REGEL 1210	82	REGEL 1710	142	REGEL 2210	79	REGEL 2710	244	REGEL 3210	199
REGEL 1220	232	REGEL 1720	224	REGEL 2220	128	REGEL 2720	201	REGEL 3220	168
REGEL 1230	21	REGEL 1730	82	REGEL 2230	107	REGEL 2730	77	REGEL 3230	114
REGEL 1240	217	REGEL 1740	195	REGEL 2240	201	REGEL 2740	255	REGEL 3240	224
REGEL 1250	124	REGEL 1750	142	REGEL 2250	180	REGEL 2750	42	REGEL 3250	244
REGEL 1260	197	REGEL 1760	5	REGEL 2260	0	REGEL 2760	89	REGEL 3260	205
REGEL 1270	196	REGEL 1770	188	REGEL 2270	226	REGEL 2770	16		
REGEL 1280	224	REGEL 1780	5	REGEL 2280	159	REGEL 2780	199		
REGEL 1290	150	REGEL 1790	53	REGEL 2290	95	REGEL 2790	176		
REGEL 1300	79	REGEL 1800	32	REGEL 2300	51	REGEL 2800	114		

Disk Base

Dit maal een inzender uit Friesland, Nick de Jong uit Leeuwarden. Hij kwam voor het probleem te staan, dat meerdere fanatieke computeraars kennen, hoe bewaren we de gegevens van de steeds maar weer groter wordende voorraad diskettes. Alle gegevens moeten toch weer snel opgezocht zijn. Hij vond de oplossing en schreef het volgende programma. Diskbase is volledig menugestuurd en dus door iedereen te begrijpen.

```
10   goto40
20   print"error:";:close15:open15,8,15
     :input#15,a,b$,c,d:printa;b$;c;d:c
     lose15
30   print"[CRSR-DOWN]toets":return
40   clr:dima$(2001):l$="[16xSPACE]":s=
     0:e=0
50   poke53280,0:poke53281,0:poke646,5:
     poke650,128
60   goto310
70   print"[SHIFT-CLR]";:ifs-1>0thengos
     ub100
80   printspc(9)"[2xCRSR-DOWN][CTRL-9]s
     top[SPACE]disk[SPACE]in[SPACE]driv
     e[SPACE]en"
90   printtab(14)"[CTRL-9]druk[SPACE]sp
     atie[CTRL-0]":wait198,197:poke198,
     0:goto170
100  print"[SHIFT-CLR]"spc(5)"[2xCRSR-D
     OWN][CTRL-9]tot[SPACE]nu[SPACE]toe
     "s;"[CRSR-LEFT][SPACE]namen[SPACE]
     ingelezen":return
110  gosub100:printspc(5)"[CTRL-9][2xSP
     ACE]nog[SPACE]een[SPACE]diskette[S
     PACE]?[SPACE](j/n)[2xSPACE]"
120  poke198,0
130  geta$:ifa$=""then130
140  ifa$="n"thenreturn
150  ifa$="j"ora$="[SPACE]"thenpoke198,
     0:goto170
160  goto120
170  printspc(5)"[SHIFT-CLR][3xCRSR-DOW
     N][CTRL-9]ik[SPACE]lees[SPACE]dire
     ctory[CTRL-0][2xCRSR-DOWN]":e=s+1:
     open1,8,0,"$":get#1,a$,b$
180  get#1,a$,b$,a$,b$:c=0:bl$="":a$(e)
     =""
```

```
190   ifa$<>""thenc=asc(a$)
200   ifb$<>""thenc=c+asc(b$)*256
210   bl$=left$(str$(c),4)
220   get#1,b$:ifst<>0thenclose1:s=e-1:f
      =0:goto110
230   ifb$<>chr$(34)then220
240   get#1,b$:ifb$<>chr$(34)thena$(e)=a
      $(e)+b$:goto240
250   get#1,b$:c$="":ifb$=chr$(32)then25
      0
260   c$=c$+b$:get#1,b$:ifb$<>""then260
270   iff=0thena$(s)=left$(a$(s)+l$,17)+
      c$:id$=mid$(a$(s),18,3):printa$(e)
      :f=1:goto300
280   ifleft$(c$,3)<>"prg"then300
290   a$(e)=left$(left$(a$(e)+l$,17)+id$
      ,19)+bl$:printa$(e):e=e+1
300   ifst=0then180
310   print"[SHIFT-CLR][CTRL-9][11xSPACE
      ]diskbase[2xSPACE]1000+[14xSPACE]"
      ;:poke198,0
320   printspc(2)"[CTRL-0][CRSR-DOWN]ges
      chreven[SPACE]door:":printspc(2)"n
      ick[SPACE]de[SPACE]jong[2xCRSR-DOW
      N]"
330   print"[5xSPACE][CTRL-9][SPACE]f1[S
      PACE][CTRL-0][SPACE]namen[SPACE]le
      zen"
340   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f2[SPACE][CTRL-0][SPACE]geh
      eugen[SPACE]legen"
350   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f3[SPACE][CTRL-0][SPACE]dat
      a[SPACE]veranderen"
360   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f4[SPACE][CTRL-0][SPACE]dat
      a[SPACE]sorteren"
370   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f5[SPACE][CTRL-0][SPACE]dat
      a[SPACE]loaden"
380   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f6[SPACE][CTRL-0][SPACE]dat
      a[SPACE]saven"
390   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f7[SPACE][CTRL-0][SPACE]dat
      a[SPACE]uitprinten"
400   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]f8[SPACE][CTRL-0][SPACE]dat
      a[SPACE]op[SPACE]scherm"
410   print"[CRSR-DOWN][5xSPACE][CTRL-9]
      [SPACE]c=[SPACE][CTRL-0][SPACE]and
      ere[SPACE]kleuren"
420   geta$:ifa$<>""then590
430   ifpeek(145)=223thenprint"[SHIFT-CL
      R][2xCRSR-DOWN]":goto450
440   goto420
450   print"[HOME][6xCRSR-DOWN]"spc(7)"[
      CTRL-9][SPACE]f1[SPACE][CTRL-0][SP
      ACE]schermrand[CRSR-DOWN]":printsp
      c(7)"[CTRL-9][SPACE]f3[SPACE][CTRL
       0][SPACE]achtergrond[CRSR-DOWN]"
460   printspc(7)"[CTRL-9][SPACE]f5[SPAC
      E][CTRL-0][SPACE]letterkleur[CRSR-
      DOWN]":printspc(7)"[CTRL-9][SPACE]
      f7[SPACE][CTRL-0][SPACE]terug[SPAC
      E]naar[SPACE]menu"
470   geta$:ifa$=""then470
480   ifa$="[F1]"then530
490   ifa$="[F3]"then550
500   ifa$="[F5]"then570
510   ifa$="[F7]"then310
520   goto470
530   a=peek(53280):a=a-239:ifa=16thena=
      0
540   poke53280,a:goto470
550   a=peek(53281):a=a-239:ifa=16thena=
      0
560   poke53281,a:goto470
570   a=peek(646):a=a+1:ifa=16thena=0
580   poke646,a:goto450
590   ifa$="[F1]"thengosub70:goto310
600   ifa$="[F2]"thenprint"[SHIFT-CLR][C
      RSR-DOWN][CRSR-RIGHT]ok":run
610   ifa$="[F3]"thengosub680:goto310
620   ifa$="[F4]"thengosub1350:goto310
630   ifa$="[F5]"thengosub1170:goto310
640   ifa$="[F6]"thengosub1260:goto310
650   ifa$="[F7]"thengosub1540:nm$="":ad
      $="":wp$="":tl$="":da$="":goto310
660   ifa$="[F8]"thengosub1390:goto310
670   goto420
680   ifs=0thenprint"[SHIFT-CLR][CRSR-DO
      WN][SPACE]er[SPACE]zijn[SPACE]geen
      [SPACE]namen[SPACE]in[SPACE]het[SP
      ACE]geheugen":print"[SPACE]druk[SP
      ACE]toets"
690   l1$=l$+l$:e=1:ifs=0thenwait198,197
      :goto310
700   l1$=l$+l$
710   print"[SHIFT-CLR][2xSPACE]element[
      SPACE]verwijderen[3xSPACE]:[SPACE]
      [CTRL-9][SPACE]`[SPACE][CTRL-0]"
720   print"[2xCRSR-DOWN][2xSPACE]elemen
      t[SPACE]verbeteren[4xSPACE]:[SPACE
      ][CTRL-9][SPACE]*[SPACE][CTRL-0]"
730   print"[2xCRSR-DOWN][2xSPACE]elemen
      t[SPACE]verder[8xSPACE]:[SPACE][CT
      RL 9][3xSPACE][CTRL-0]"
740   print"[2xCRSR-DOWN][2xSPACE]elemen
      t[SPACE]terug[9xSPACE]:[SPACE][CTR
      L 9][SPACE][kwadraatpijl][SPACE][C
      TRL 0]"
750   print"[2xCRSR-DOWN][2xSPACE]nieuw[
      SPACE]element[9xSPACE]:[SPACE][CTR
      L 9][SPACE]+[SPACE][CTRL-0]"
760   print"[2xCRSR-DOWN][2xSPACE]elemen
      t[SPACE]zoeken[8xSPACE]:[SPACE][CT
      RL 9][SPACE]|[SPACE][CTRL-0]"
770   print"[2xCRSR-DOWN][2xSPACE]terug[
      SPACE]naar[SPACE]menu[7xSPACE]:[SP
      ACE][CTRL-9][SPACE][pijl links][SP
      ACE][CTRL-0]":goto790
780   ife=1ore=sthenprint"[CRSR-UP][6xSP
      ACE]"
790   print"[HOME][22xCRSR-DOWN]";:e$=st
      r$(e):print"("right$(e$,len(e$)-1)
      ")[SPACE]"
800   print"[CRSR-UP]";:printspc(6)a$(e)
      ;:q=33-len(a$(e)):printleft$(l1$,q
      )
810   geta$:ifa$=""then810
820   ifa$="[SPACE]"thene=e+1:ife>sthene
      =1
830   ifa$="[kwadraatpijl]"thene=e-1:ife
      <1thene=s
840   ifa$="`"then900
850   ifa$="*"then910
860   ifa$="+"then930
870   ifa$="[pijl links]"then310
880   ifa$="|"then960
```

```
890   goto780
900   forq=etos:a$(q)=a$(q+1):next:s=s-1
      :goto780
910   print"[HOME][21xCRSR-DOWN][CRSR-RI
      GHT][3xSPACE]korrektie:[2xSPACE]":
      input"[4xSPACE]";a$(e)
920   print"[HOME][21xCRSR-DOWN][18xSPAC
      E]":print"[6xSPACE]":goto780
930   print"[HOME][21xCRSR-DOWN][CRSR-RI
      GHT][3xSPACE]nieuw[SPACE]element":
      input"[4xSPACE]";a$(s+1):s=s+1
940   a$(s)=left$(a$(s),23)
950   goto920
960   print"[SHIFT-CLR][CRSR-DOWN][CTRL
      9][2xSPACE]element[SPACE]zoeken[SP
      ACE]:[2xSPACE][CTRL-0][3xCRSR-DOWN
      ]"
970   print"[SPACE][CTRL-9][SPACE]1[SPAC
      E][CTRL-0][SPACE]-[SPACE]op[SPACE]
      naam[SPACE](*[SPACE]is[SPACE]toege
      staan)"
980   print"[CRSR-DOWN][SPACE][CTRL-9][S
      PACE]2[SPACE][CTRL-0][SPACE]-[SPAC
      E]op[SPACE]disknummer"
990   print"[CRSR-DOWN][SPACE][CTRL-9][S
      PACE][pijl links][SPACE][CTRL-0][S
      PACE]-[SPACE]terug[SPACE]naar[SPAC
      E]menu"
1000  geta$:ifa$=""then1000
1010  ifa$="[pijl links]"then710
1020  ifa$="1"then1050
1030  ifa$="2"then1120
1040  goto1000
1050  poke19,64:input"[HOME][13xCRSR-DOW
      N]naam:";n$:poke19,0:n$=left$(n$,1
      6):print
1060  l=len(n$):ifmid$(n$,l,1)="*"thenl=
      l-1:n$=left$(n$,l):w=0:goto1100
1070  n$=n$+l$:n$=left$(n$,16)
1080  forq=0tos:ifleft$(a$(q),16)=n$then
      printa$(q):w=w+1:ifw=23thengosub11
      60
1090  next:print"[CRSR-DOWN][CTRL-9]eind
      e[SPACE]lijst.[2xSPACE]druk[SPACE]
      toets[CTRL-0]":wait198,197:goto960
1100  forq=0tos:ifleft$(a$(q),l)=n$thenp
      rinta$(q):w=w+1:ifw=23thengosub116
      0
1110  next:print"[CRSR-DOWN][CTRL-9]eind
      e[SPACE]lijst.[2xSPACE]druk[SPACE]
      toets[CTRL-0]":wait198,197:goto960
1120  poke19,64:input"[HOME][13xCRSR-DOW
      N]nummer:";n$:poke19,0
1130  n$=n$+"[3xSPACE]":n$=left$(n$,3):p
      rint
1140  forq=0tos:ifmid$(a$(q),18,3)=n$the
      nprinta$(q)
1150  next:print"[CRSR-DOWN][CTRL-9]eind
      e[SPACE]lijst.[2xSPACE]druk[SPACE]
      toets[CTRL-0]":wait198,197:goto960
1160  poke198,0:wait198,197:w=0:poke198,
      0:return
1170  input"[SHIFT-CLR][2xCRSR-DOWN]naam
      [SPACE]file:";m$:ifleft$(m$,1)<>"*
      "thenm$=left$(m$,16):goto1190
1180  goto1170
1190  open2,8,2,m$:e=1
1200  n$="":input#2,n$:an=val(n$):ifan<1
      thenclose2:gosub20:wait198,197:ret
      urn
1210  print"[CRSR-DOWN]in[SPACE]te[SPACE
      ]lezen[SPACE]namen";an
1220  print"[CRSR-DOWN]naam:[SPACE]";
1230  ife>anthenclose2:s=s+(e-1):gosub13
      30:return
1240  input#2,a$(s+e)
1250  printtab(7)e"[CRSR-UP]":e=e+1:goto
      1230
1260  input"[SHIFT-CLR][2xCRSR-DOWN]naam
      [SPACE]te[SPACE]saven[SPACE]file:"
      ;m$:m$=left$(m$,16)
1270  open2,8,2,"0:"+m$+",s,w"
1280  print#2,str$(s):print"aantal:[SPAC
      E]"s
1290  print"[CRSR-DOWN]naam:[SPACE]";:q=
      1
1300  ifq>sthen1320
1310  print#2,a$(q):printtab(7)q"[CRSR-U
      P]":q=q+1:goto1300
1320  close2:return
1330  close15:open15,8,15:input#15,a,b$,
      c,d:ifa=0thenreturn
1340  print"[2xCRSR-DOWN][CTRL-9][SPACE]
      disk[SPACE]error:[SPACE][CTRL-0]"a
      ;b$;c;d:wait198,197:close15:return
1350  remifpeek(49152)<>76andpeek(49153)
      <>100thenprint"[SHIFT-CLR]geen[SPA
      CE]sorter":end
1360  print"[SHIFT-CLR][CRSR-DOWN][CTRL
      9]ben[SPACE]aan[SPACE]het[SPACE]so
      rteren":ti$="000000":print"[2xCRSR
      -DOWN]"s"woorden[SPACE]";
1370  sys49152,s,a$(1):print"in"ti/60"se
      c"
1380  poke198,0:print"[2xCRSR-DOWN][CTRL
       9]druk[SPACE]toets[CTRL-0]":wait1
      98,197:return
1390  print"[SHIFT-CLR][2xCRSR-DOWN][2xC
      RSR-RIGHT][CTRL-9][SPACE][pijl lin
      ks][SPACE]=[SPACE]menu[SPACE][CTRL
      0]":lt=0:w=0
1400  printtab(7)a$(lt):w=w+1:lt=lt+1:if
      w=23thengosub1460:ifa$="[pijl link
      s]"then1450
1410  geta$:ifa$="[pijl links]"then1450
1420  iflt<s+1then1400
1430  print"[CRSR-DOWN][CTRL-9][SPACE]dr
      uk[SPACE]op[SPACE][pijl links][SPA
      CE][CTRL-0]"
1440  geta$:ifa$<>"[pijl links]"then1440
1450  return
1460  geta$:ifa$=""then1460
1470  w=0:return
1480  printchr$(147)
1490  open1,4,0:close1:ifst=0thenreturn
1500  print"[HOME][10xCRSR-DOWN][CTRL-9]
      [SPACE]error:[2xSPACE]device[SPACE
      ]4[SPACE]not[SPACE]present"
1510  print"[2xCRSR-DOWN][CTRL-9][SPACE]
      doe[SPACE]printer[SPACE]aan[SPACE]
      of[SPACE]druk[SPACE]'[pijl links]'
      "
1520  geta$:ifa$="[pijl links]"thenretur
      n
1530  goto1490
1540  gosub1480:ifa$<>"[pijl links]"then
      print"[SHIFT-CLR][CTRL-9]m[CTRL-0]
      et[SPACE]of[SPACE][CTRL-9]z[CTRL-0
      ]onder[SPACE]header":goto1560
1550  return
```

```
1560   geta$:ifa$=""then1560
1570   ifa$="m"then1600
1580   ifa$="z"then1640
1590   goto1550
1600   input"[SHIFT-CLR][2xCRSR-DOWN]soor
       t";so$:print:so$=left$(so$,35):inp
       ut"naam";nm$
1610   nm$=left$(nm$,35):print:input"adre
       s";ad$:ad$=left$(ad$,35):print
1620   input"woonplaats";wp$:wp$=left$(wp
       $,35):print:input"tel";tl$:print
1630   tl$=left$(tl$,35):input"datum";da$
       :da$=left$(da$,35):print
1640   input"aantal[SPACE]regels[SPACE]pe
       r[SPACE]blad[SPACE]";ar
1650   ifar<1thenar=1
1660   ifar>2000thenar=2000
1670   open4,4:re$=chr$(13):gr$=chr$(14):
       kl$=chr$(15):vn$=chr$(18):vo$=chr$
       (146)
1680   ifa$="z"then1750
1690   print#4,gr$;re$;">>[SPACE]"so$"[SP
       ACE]<<";re$:ifnm$<>""thenprint#4,n
       m$
1700   ifad$<>""thenprint#4,ad$
1710   iftl$<>""thenprint#4,tl$
1720   ifwp$<>""thenprint#4,wp$
1730   ifda$<>""thenprint#4,re$;"diskette
       [SPACE]inhoud[SPACE]van[SPACE]"da$
       ;re$;
1740   print#4,re$;"programma's:"s;re$
1750   ib$="prg.naam[9xSPACE]id[SPACE]bl[
       4xSPACE]":print#4,kl$;ib$;ib$;ib$
1760   sr$="":forq=1to75:sr$=sr$+"[SHIFT
       *]":next:print#4,sr$;re$
1770   nr=0:an=int(s/3)+1:a1=1:r=0
1780   p1=26-len(a$(a1)):p2=26-len(a$(a1+
       an)):r=r+1:al=al+1
1790   print#4,a$(a1);spc(p1);a$(a1+an);s
       pc(p2);a$(a1+(2*an)):a1=a1+1
1800   ifa1=an+1thenprint"klaar":wait198,
       197:el=0:al=0:close4:return
1810   print"[SHIFT-CLR][2xCRSR-DOWN]nu[S
       PACE]"al"[SPACE]regels[SPACE]gepri
       nt"
1820   ifr=arthengoto1840:r=0
1830   goto1780
1840   print"[4xCRSR-DOWN][CTRL-9][4xSPAC
       E][CTRL-0][SPACE]doorgaan":print"[
       CRSR-DOWN][CTRL-9][SPACE]f1[SPACE]
       [CTRL-0][SPACE]prg.naam,[SPACE]id,
       [SPACE]bl[SPACE]en[SPACE]lijn[SPAC
       E]"
1850   print"[CRSR-DOWN][CTRL-9][SPACE]f3
       [SPACE][CTRL-0][SPACE]een[SPACE]re
       gel[SPACE]printen":print"[CRSR-DOW
       N][CTRL-9][SPACE]f5[SPACE][CTRL-0]
       [SPACE]nieuw[SPACE]aantal[SPACE]re
       gels[SPACE]ingeven"
1860   print"[CRSR-DOWN][CTRL-9][SPACE]f6
       [SPACE][CTRL-0][SPACE]header,lijn[
       SPACE]en[SPACE]doorgaan":print"[CR
       SR-DOWN][CTRL-9][SPACE]f7[SPACE][C
       TRL 0][SPACE]stoppen"
1870   geta$:ifa$=""then1870
1880   ifa$=chr$(32)thenr=0:goto1830
1890   ifasc(a$)<133orasc(a$)>139then1870
1900   q=asc(a$)-132:onqgoto1920,1930,194
       0,1910,1870,1870,1690,1870
1910   close4:al=0:ar=0:r=0:el=0:return
1920   print#4,kl$;ib$;ib$;ib$;re$;sr$;re
       $:goto1870
1930   r=ar-1:goto1780
1940   input"aantal[SPACE]regels[SPACE]pe
       r[SPACE]keer";ar
1950   ifar<1thenar=1
1960   ifar>2000thenar=2000
1970   print"[SHIFT-CLR][2xCRSR-DOWN]nu[S
       PACE]"al"[SPACE]regels[SPACE]gepri
       nt":goto1840
```

** EINDE LISTING disk-bas **

REGEL	10	237	REGEL	330	229	REGEL	650	83	REGEL	970	157	REGEL	1290	238
REGEL	20	69	REGEL	340	189	REGEL	660	202	REGEL	980	143	REGEL	1300	77
REGEL	30	69	REGEL	350	15	REGEL	670	31	REGEL	990	252	REGEL	1310	233
REGEL	40	69	REGEL	360	152	REGEL	680	228	REGEL	1000	142	REGEL	1320	154
REGEL	50	42	REGEL	370	218	REGEL	690	56	REGEL	1010	132	REGEL	1330	90
REGEL	60	29	REGEL	380	165	REGEL	700	221	REGEL	1020	132	REGEL	1340	180
REGEL	70	69	REGEL	390	59	REGEL	710	221	REGEL	1030	131	REGEL	1350	124
REGEL	80	46	REGEL	400	139	REGEL	720	3	REGEL	1040	74	REGEL	1360	197
REGEL	90	19	REGEL	410	215	REGEL	730	175	REGEL	1050	116	REGEL	1370	178
REGEL	100	181	REGEL	420	29	REGEL	740	164	REGEL	1060	241	REGEL	1380	58
REGEL	110	187	REGEL	430	22	REGEL	750	154	REGEL	1070	44	REGEL	1390	112
REGEL	120	149	REGEL	440	31	REGEL	760	15	REGEL	1080	50	REGEL	1400	144
REGEL	130	97	REGEL	450	47	REGEL	770	125	REGEL	1090	201	REGEL	1410	246
REGEL	140	105	REGEL	460	130	REGEL	780	194	REGEL	1100	23	REGEL	1420	120
REGEL	150	210	REGEL	470	104	REGEL	790	4	REGEL	1110	201	REGEL	1430	198
REGEL	160	28	REGEL	480	170	REGEL	800	30	REGEL	1120	220	REGEL	1440	167
REGEL	170	238	REGEL	490	173	REGEL	810	102	REGEL	1130	159	REGEL	1450	142
REGEL	180	167	REGEL	500	176	REGEL	820	129	REGEL	1140	241	REGEL	1460	152
REGEL	190	176	REGEL	510	169	REGEL	830	186	REGEL	1150	201	REGEL	1470	1
REGEL	200	232	REGEL	520	36	REGEL	840	102	REGEL	1160	218	REGEL	1480	77
REGEL	210	53	REGEL	530	168	REGEL	850	81	REGEL	1170	148	REGEL	1490	26
REGEL	220	22	REGEL	540	100	REGEL	860	84	REGEL	1180	82	REGEL	1500	102
REGEL	230	15	REGEL	550	169	REGEL	870	128	REGEL	1190	146	REGEL	1510	85
REGEL	240	77	REGEL	560	101	REGEL	880	136	REGEL	1200	214	REGEL	1520	186
REGEL	250	182	REGEL	570	216	REGEL	890	40	REGEL	1210	213	REGEL	1530	87
REGEL	260	99	REGEL	580	0	REGEL	900	33	REGEL	1220	128	REGEL	1540	111
REGEL	270	50	REGEL	590	93	REGEL	910	149	REGEL	1230	141	REGEL	1550	142
REGEL	280	53	REGEL	600	18	REGEL	920	206	REGEL	1240	218	REGEL	1560	153
REGEL	290	244	REGEL	610	149	REGEL	930	147	REGEL	1250	144	REGEL	1570	161
REGEL	300	84	REGEL	620	196	REGEL	940	110	REGEL	1260	49	REGEL	1580	178
REGEL	310	190	REGEL	630	193	REGEL	950	36	REGEL	1270	120	REGEL	1590	84
REGEL	320	239	REGEL	640	197	REGEL	960	104	REGEL	1280	179	REGEL	1600	40

REGEL 1610	196	REGEL 1690	168	REGEL 1770	14	REGEL 1850	192	REGEL 1930	6
REGEL 1620	147	REGEL 1700	36	REGEL 1780	196	REGEL 1860	96	REGEL 1940	24
REGEL 1630	7	REGEL 1710	90	REGEL 1790	73	REGEL 1870	157	REGEL 1950	31
REGEL 1640	4	REGEL 1720	104	REGEL 1800	111	REGEL 1880	137	REGEL 1960	63
REGEL 1650	31	REGEL 1730	152	REGEL 1810	225	REGEL 1890	66	REGEL 1970	11
REGEL 1660	63	REGEL 1740	213	REGEL 1820	141	REGEL 1900	11		
REGEL 1670	62	REGEL 1750	159	REGEL 1830	89	REGEL 1910	15		
REGEL 1680	180	REGEL 1760	151	REGEL 1840	186	REGEL 1920	244		

Floppy

Floppy is geschreven door Dick Sietses uit s'Gravenzande. Met dit programma kunnen de belangrijkste funkties van de commodore 1541 Disk Drive op een eenvoudige wijze worden uitgevoerd.

```
10   rem * c *    f l o p p y    * c *
20   rem * 6 *  door dick sietses  * 6 *
30   rem * 4 * uit  's-gravenzande * 4 *
40   rem
50   poke53280,4:poke53281,4
60   print"[SHIFT-CLR]"
70   print"[CRSR-DOWN][CTRL-8][13xSPACE]m[SPACE]e[SPACE]n[SPACE]u[SPACE]"
80   print"[4xCRSR-DOWN][CTRL-2][CTRL-9][COM-K]1[CTRL-0][COM-K][SPACE]formatteren"
90   print"[CRSR-DOWN][CTRL-9][COM-K]2[CTRL-0][COM-K][SPACE]verwijderen[SPACE]van[SPACE]een[SPACE]programma"
100  print"[CRSR-DOWN][CTRL-9][COM-K]3[CTRL-0][COM-K][SPACE]inhoud[SPACE]disk[SPACE]lezen[SPACE]"
110  print"[CRSR-DOWN][CTRL-9][COM-K]4[CTRL-0][COM-K][SPACE]programma[SPACE]nieuwe[SPACE]naam[SPACE]geven"
120  print"[CRSR-DOWN][CTRL-9][COM-K]5[CTRL-0][COM-K][SPACE]einde"
130  print"[CRSR-DOWN][CTRL-9][COM-K]6[CTRL-0][COM-K][SPACE]lege[SPACE]ruimten[SPACE]tussen[SPACE]de[SPACE]programma's[6xSPACE]wegwerken"
140  print"[3xCRSR-DOWN][SPACE]e[SPACE]v[SPACE]e[SPACE]n[3xSPACE]g[SPACE]e[SPACE]d[SPACE]u[SPACE]l[SPACE]d[SPACE]![SHIFT-SPACE]![CRSR-UP]":close1
150  print"[26xSPACE][4xCRSR-UP]":close1:gosub380
160  print"[2xCRSR-DOWN]uw[SPACE]keuze[CTRL-1][2xCRSR-DOWN][6xCRSR-LEFT][CRSR-DOWN][CRSR-UP]";
170  poke204,0:geta$:ifa$=""then170
180  poke204,1:onval(a$)goto200,230,240,290,450,310
190  goto170
200  input"[SHIFT-CLR][CRSR-DOWN]naam[SPACE]diskette";a$
210  input"nummer[SPACE]diskette";b$:b$=str$(val(b$))
220  a$="new:"+a$+","+b$:open1,8,15,a$:save"floppy",8:goto60
230  input"welk[SPACE]programma";a$:a$="scratch:"+a$:open1,8,15,a$:goto60
240  gosub320:print"[SHIFT-CLR]diskette[SPACE]naam:";:open1,8,0,a$:get#1,a$,a$
250  get#1,a$,a$:ifa$=""thenclose1:input"<return>";c$:x$="":goto60
260  get#1,a$,b$:bf=asc(a$+chr$(0))+256*asc(b$+chr$(0))
270  get#1,a$:ifa$=""thengosub390:goto250
280  x$=x$+a$:goto270
290  input"oude[SPACE]naam";a$:input"nieuwe[SPACE]naam";b$:a$="r:"+b$+"="+a$:open1,8,15,a$
300  goto60
310  open1,8,15,"validate":goto60
320  print:print"[CRSR-UP][SHIFT-CLR]wilt[SPACE]u[SPACE]de[SPACE]directory[SPACE]van:[CTRL-9][COM-K]1[CTRL-0][COM-K]de[SPACE]hele[SPACE]disk":printtab(24);
330  print"[CTRL-9][COM-K]2[CTRL-0][COM-K]een[SPACE]gedeelte":printtab(27);"van[SPACE]de[SPACE]disk":inputa$:a=val(a$)
340  a$="$":ifa=1thenreturn
350  print"van[SPACE]welk[SPACE]gedeelte(voor[SPACE]willekeurige[SPACE]let-ter[SPACE]een'?'.";
360  print"en[2xSPACE]de[2xSPACE]rest[2xSPACE]van[2xSPACE]het[SPACE]woord[SPACE]willekeurig[SPACE]een'*'"
370  input"welk[SPACE]gedeelte";a$:a$="$:"+a$:return
380  open1,8,15:input#1,a$,b$,c$,d$:print"situatie[SPACE]disk:[SPACE]"b$:close1:return
390  a=1:ifleft$(x$,12)="blocks[SPACE]free."thenprint"vrij[SPACE]geheugen:"bf;"blocks":return
400  a=a+1:ifmid$(x$,a,1)=chr$(34)thenv$=right$(x$,((len(x$))-a)):goto420
410  goto400
420  v$=left$(v$,16)
430  printchr$(34);
440  printv$;bf:x$="":return
450  new
```

** EINDE LISTING floppy64 **

REGEL 10	151	REGEL 100	119	REGEL 190	33	REGEL 280	21	REGEL 370	165
REGEL 20	18	REGEL 110	13	REGEL 200	135	REGEL 290	61	REGEL 380	28
REGEL 30	109	REGEL 120	110	REGEL 210	126	REGEL 300	239	REGEL 390	223
REGEL 40	143	REGEL 130	133	REGEL 220	194	REGEL 310	169	REGEL 400	114
REGEL 50	45	REGEL 140	113	REGEL 230	226	REGEL 320	239	REGEL 410	29
REGEL 60	112	REGEL 150	142	REGEL 240	103	REGEL 330	248	REGEL 420	82
REGEL 70	193	REGEL 160	108	REGEL 250	131	REGEL 340	157	REGEL 430	83
REGEL 80	132	REGEL 170	40	REGEL 260	92	REGEL 350	246	REGEL 440	74
REGEL 90	174	REGEL 180	186	REGEL 270	208	REGEL 360	28	REGEL 450	16

Flash

Edwin van der Elst uit Rotterdam stuurde ons dit programma. Het beeld snel laten knipperen konden we al wel. Maar nu zonder balken over het beeld te laten lopen. Uiteraard weer geen interrupt.

```
10    rem in fl$ de tekst die moet knipp
      eren,in h het hoogste cijfer dat i
      ngetoetst
20    rem kan worden,in t het aantal spa
      ties
30    rem voorbeeld:
40    printchr$(147):fl$="commodore[SPAC
      E]info[2xSPACE](1-3)":t=9:h=3
50    gosub 90:rem aanroepen flash-sub
                          routine
60    :
70    :
80    end
90    fl$="[CTRL-9]"+fl$
100   printtab(t);fl$:fora=1to60
110   geti$:ifi$<>""then150
120   next:print"[CRSR-UP]";tab(t);fl$:f
      ora=1to60
130   geti$:ifi$<>""then150
140   next:print"[CRSR-UP]";:goto100
150   ifval(i$)<1orval(i$)>hthenprint"[C
      RSR-UP]";:goto100
160   i=val(i$):return
```

** EINDE LISTING flash-64 **

REGEL	10	58	REGEL	90	3
REGEL	20	250	REGEL	100	147
REGEL	30	107	REGEL	110	37
REGEL	40	198	REGEL	120	68
REGEL	50	3	REGEL	130	37
REGEL	60	58	REGEL	140	185
REGEL	70	58	REGEL	150	194
REGEL	80	128	REGEL	160	70

Orgel 128

Deze inzender, Y. Rozijn, kennen we ook zo langzamerhand wel als trouwe inzender van prima 128 programma's, dit maal met het programma orgel. Met dit programma maak men van de 128 een orgeltje. Het toetsenbord fungeert als klavier. De verdeling van de witte en zwarte toetsen wordt op het scherm getoond. Akkoorden spelen kan ook, door twee of drie toetsen tegelijk in te drukken. Met de funktietoetsen kun je de waveform kiezen.

```
10    rem ******************************
20    rem **          orgel           **
30    rem **        voor c-128        **
40    rem ** door y rozijn, amsterdam **
50    rem ******************************
60    color 0,16:color 4,16:color 5,7:co
      lor 6,16:scnclr:print chr$(11)chr$
      (14);
70    char 1,12,0,"O[2xSPACE]R[2xSPACE]G
      [2xSPACE]E[2xSPACE]L"
80    char 1,0,3,"Indeling[SPACE]van[SPA
      CE]de[SPACE]witte[SPACE]en[SPACE]z
      warte[SPACE]toetsen:"
90    char 1,9,11,"[kwadraatpijl]":char
      1,0,12,"De[SPACE]linker[SPACE]SHIF
      T[SPACE]is[SPACE]de[SPACE]centrale
      [SPACE]C"
100   color 5,1:char 1,2,5,"1[CRSR-RIGHT
      ]2[3xCRSR-RIGHT]4[CRSR-RIGHT]5[CRS
      R-RIGHT]6[3xCRSR-RIGHT]8[CRSR-RIGH
      T]9[3xCRSR-RIGHT]+[CRSR-RIGHT]-[CR
      SR-RIGHT]|",1
110   char 1,10,9,"a[CRSR-RIGHT]s[3xCRSR
      -RIGHT]f[CRSR-RIGHT]g[CRSR-RIGHT]h
      [3xCRSR-RIGHT]k[CRSR-RIGHT]l[3xCRS
      R-RIGHT];[CRSR-RIGHT]=[CRSR-RIGHT]
      R",1
120   color 5,2:for i=1 to 27 step 2:cha
      r 1,i,5,"[SPACE]",1:char 1,i+8,9,"
      [SPACE]",1:next
130   char 1,1,6,"C[CRSR-RIGHT]q[CRSR-RI
      GHT]w[CRSR-RIGHT]e[CRSR-RIGHT]r[CR
      SR-RIGHT]t[CRSR-RIGHT]y[CRSR-RIGHT
      ]u[CRSR-RIGHT]i[CRSR-RIGHT]o[CRSR-
      RIGHT]p[CRSR-RIGHT]`[CRSR-RIGHT]*[
      CRSR-RIGHT][kwadraatpijl]",1
140   char 1,9,10,"S[CRSR-RIGHT]z[CRSR-R
      IGHT]x[CRSR-RIGHT]c[CRSR-RIGHT]v[C
      RSR-RIGHT]b[CRSR-RIGHT]n[CRSR-RIGH
      T]m[CRSR-RIGHT],[CRSR-RIGHT].[CRSR
      -RIGHT]/[CRSR-RIGHT]S[CRSR-RIGHT]C
      [CRSR-RIGHT]C",1
150   color 5,7:char 1,0,15,"Waveform[SP
      ACE]instellen:[SPACE]F1,F3,F5,F7"
160   char 1,0,19,"Even[SPACE]wachten...
      "
170   rem keyboard scan routine
180   bank 15:for i=0to 73:read k:poke
      dec("1300")+i,k:next
190   data 120,169,255,141,47,208,169,25
      4
200   data 162,0,134,250,133,252,141,0
210   data 220,173,1,220,205,1,220,208
220   data 248,160,7,42,176,21,72,132
230   data 251,165,250,10,10,10,5,251
240   data 168,185,0,20,157,0,11,232
250   data 164,251,104,136,16,229,230,25
      0
260   data 165,252,56,42,176,206,169,50
270   data 157,0,11,157,1,11,157,2
280   data 11,96
290   rem toets/noot tabel
300   for i=0to 63:read k:poke dec("140
      0")+i,k:next
310   data 50,46,47,54,51,52,53,45
320   data 50,4,25,6,26,27,5,24
330   data 8,7,50,10,29,30,9,28
340   data 50,11,32,13,33,34,12,31
350   data 15,14,50,50,36,37,16,35
360   data 18,17,39,20,40,50,19,38
370   data 22,21,42,50,43,44,23,41
380   data 1,55,0,3,50,50,2,56
390   rem sid frekwenties
400   dim f(50):for i=36 to 47:read f(i)
      :next
410   data 8912,9442,10004,10599,11216,1
      1897
420   data 12604,13353,14148,14989,15880
      ,16824
430   k=1:for s=2 to 0step -1:k=k*2
440   : for i=0to 11:f(12*s+i)=f(36+i)/
```

```
      k:next
450   next
460   k=dec("1300"):b=dec("0b00")-1:l=32
      767:w=2
470   poke dec("0328"),112:trap 630
480   char 1,0,19,"Stoppen[SPACE]dmv[SPA
      CE]de[SPACE]STOP[SPACE]toets"
490   if rwindow(2)=40then begin
500   : char 1,0,22,"Druk[SPACE]een[SPAC
      E]toets.[SPACE]Daarna[SPACE]wordt[
      SPACE]het[SPACE]scherm"
510   : char 1,0,23,"grijs[SPACE]en[SPAC
      E]kunt[SPACE]u[SPACE]beginnen[SPAC
      E]te[SPACE]spelen."
520   : char 1,0,24,"Met[SPACE][pijl[SPA
      CE]links][SPACE]haalt[SPACE]u[SPAC
      E]dit[SPACE]scherm[SPACE]even[SPAC
      E]terug."
530   : wait 212,255,88
540   bend
550   fast
560   do:sysk:fors=1to3:i=peek(b+s):ifi>
      50thengosub590
570   f=f(i):ifs(s)-fthensounds,.,.:soun
      ds,f,l,,,,w:s(s)=f
580   next:loop
590   if i<55 then w=i-51
600   if i=55 then slow:do:sysk:loop whi
      le peek(b+1)=55:fast
610   if i=56 then sysdec("fa53"):end
620   i=50:return
630   slow:print:print err$(er)"[SPACE]e
      rror[SPACE]in"el
640   for s=1 to 3:sound s,0,0:next
650   poke 208,0
660   end
```

** EINDE LISTING orgel128 **

REGEL 10	123	REGEL 350	238
REGEL 20	176	REGEL 360	244
REGEL 30	136	REGEL 370	226
REGEL 40	106	REGEL 380	28
REGEL 50	123	REGEL 390	182
REGEL 60	92	REGEL 400	13
REGEL 70	101	REGEL 410	8
REGEL 80	205	REGEL 420	119
REGEL 90	82	REGEL 430	217
REGEL 100	42	REGEL 440	107
REGEL 110	205	REGEL 450	130
REGEL 120	199	REGEL 460	31
REGEL 130	162	REGEL 470	52
REGEL 140	38	REGEL 480	97
REGEL 150	99	REGEL 490	184
REGEL 160	181	REGEL 500	76
REGEL 170	43	REGEL 510	37
REGEL 180	135	REGEL 520	188
REGEL 190	92	REGEL 530	197
REGEL 200	165	REGEL 540	23
REGEL 210	161	REGEL 550	35
REGEL 220	137	REGEL 560	83
REGEL 230	114	REGEL 570	142
REGEL 240	76	REGEL 580	168
REGEL 250	73	REGEL 590	251
REGEL 260	248	REGEL 600	207
REGEL 270	229	REGEL 610	69
REGEL 280	128	REGEL 620	40
REGEL 290	245	REGEL 630	123
REGEL 300	231	REGEL 640	105
REGEL 310	248	REGEL 650	141
REGEL 320	89	REGEL 660	128
REGEL 330	93		
REGEL 340	219		

IRQ in Basic 128

Van dezelfde inzender Y. Rozijn het volgende programma IRQ in Basic. Met programmering via de IRQ interrupt zijn fraaie effecten te bereiken. Helaas was dit alleen weggelegd voor de machinetaal programmeurs. Maar met een eenvoudige truc is iets dergelijks ook mogelijk in Basic. In het programma IRQ in Basic staat een uitvoerige uitleg, het geeft bovendien een voorbeeld van een interrupt routine. Het één en ander is goed in een eigen programma te gebruiken.

```
1     rem ******************************
2     rem **       irq in basic       **
3     rem **        voor c-128        **
4     rem ** door y rozijn, amsterdam **
5     rem ******************************
100   rem  periodiek uitvoeren van een
110   rem  interrupt routine in basic
120   rem  gaat op de volgende manier:
130   rem  1. poke deze machinetaal
140   rem     routine weg:
150   :
160   for i=0to 57:read q:poke dec("130
      0")+i,q:next
170   data 120,141,58,19,141,59,19,170
180   data 240,14,173,127,18,41,251,141
190   data 127,18,169,39,162,19,208,7
200   data 141,120,18,169,101,162,250,14
      1
210   data 20,3,142,21,3,88,96,206
220   data 58,19,208,11,173,59,19,141
230   data 58,19,169,255,141,120,18,76
240   data 101,250
250   :
260   rem  2. meld de interrupt routine
270   rem     aan als collision routine
280   rem     voor type 3 (lightpen):
290   :
300   collision 3,730
310   :
320   rem  3. aktiveer de machinetaal-
330   rem     routine. geef als paramete
      r
340   rem     op na hoeveel jiffies (ca
350   rem     1/50sec elk) de interrupt
360   rem     routine moet worden uitge-
370   rem     voerd (0=stop).
380   :
390   sysdec("1300"),10
400   :
410   rem  nu wordt ongeveer elke 1/5 se
      c.
420   rem  (10jiffies) de routine op
430   rem  regel 730uitgevoerd.
440   rem
450   rem  ondertussen kun je gewoon met
460   rem  je programma verdergaan.
470   rem  bijvoorbeeld:
480   :
490   rem definieer sprite 1 als blok:
500   for a=3584 to 3646:poke a,255:next
510   i=90:movespr 1,180,60:sprite 1,1,2
520   print "de[SPACE]sprite[SPACE]bewee
      gt[SPACE]in[SPACE]een[SPACE]cirkel
      "
530   print:print "stop[SPACE]met[SPACE]
      druk[SPACE]op[SPACE]toets"
540   do
```

```
550     : x=x+1:print x;
560     : get x$
570     loop while x$=""
580     :
590     rem  uitzetten van de interrupt
600     rem  routine als volgt:
610     :
620     sysdec("1300"),0
630     :
640     print "routine[SPACE]uit[SPACE]->[
        SHIFT-SPACE]sprite[SPACE]gaat[SPAC
        E]rechtdoor"
650     end
660     :
670     rem  onderstaande routine zorgt
680     rem  ervoor dat bij elke aanroep
690     rem  (ca elke 1/5 sekonde) de spri
        te
700     rem  van richting verandert zodat
710     rem  hij in een cirkel beweegt.
720     :
730     i=i+3:if i>360then i=i-360
740     movespr 1,i#1:return
750     :
760     rem  zet de routine bij voorkeur z
        o
770     rem  ver mogelijk vooraan in het
780     rem  programma. dat scheelt tijd!
```

** EINDE LISTING irqbasic **

REGEL 1	123	REGEL 480	58
REGEL 2	28	REGEL 490	100
REGEL 3	136	REGEL 500	85
REGEL 4	106	REGEL 510	85
REGEL 5	123	REGEL 520	105
REGEL 100	169	REGEL 530	64
REGEL 110	123	REGEL 540	235
REGEL 120	30	REGEL 550	221
REGEL 130	92	REGEL 560	87
REGEL 140	210	REGEL 570	91
REGEL 150	58	REGEL 580	58
REGEL 160	245	REGEL 590	150
REGEL 170	189	REGEL 600	91
REGEL 180	227	REGEL 610	58
REGEL 190	154	REGEL 620	36
REGEL 200	65	REGEL 630	58
REGEL 210	240	REGEL 640	158
REGEL 220	147	REGEL 650	128
REGEL 230	201	REGEL 660	58
REGEL 240	216	REGEL 670	195
REGEL 250	58	REGEL 680	65
REGEL 260	141	REGEL 690	131
REGEL 270	17	REGEL 700	249
REGEL 280	48	REGEL 710	196
REGEL 290	58	REGEL 720	58
REGEL 300	14	REGEL 730	168
REGEL 310	58	REGEL 740	198
REGEL 320	24	REGEL 750	58
REGEL 330	123	REGEL 760	44
REGEL 340	113	REGEL 770	92
REGEL 350	138	REGEL 780	144
REGEL 360	100		
REGEL 370	65		
REGEL 380	58		
REGEL 390	85		
REGEL 400	58		
REGEL 410	220		
REGEL 420	143		
REGEL 430	196		
REGEL 440	143		
REGEL 450	27		
REGEL 460	209		
REGEL 470	64		

Cardes c 16

Steven Bakker uit Wieringerwerf. Character Designer heet zijn programma. Hij geeeft u hiermee de mogelijkheid zelf karakters te maken, deze weg te schrijven en dan later in een eigen programma te gebruiken. Hiermee is het dan mogelijk bijvoorbeeld een sierletter op het beeldscherm te schrijven. Nadat u de characters heeft gemaakt drukt u op Esc, dit mag u pas doen als u helemaal klaar bent want het programma vernietigd zichzelf. Er blijft een basicloader over die zonder aanpassingen in een eigen programma kan worden gebruikt. Het is voldoende deze op te roepen aan het begin van uw eigen programma.

```
10      goto1050
20      :
30      rem-----cursor+uitlezing keyboard-
        --
40      cp=3354+mn+y*40+x:fs=cp-1024
50      ifpeek(cp)<128thenpokecp,peek(cp)+
        128:pokefs,peek(fs)or128
60      poke239,0:getkeyt$:sound3,th,1:pok
        ecp,peek(cp)-128:pokefs,peek(fs)an
        d127:return
70      :
80      rem-------------------------------
        --
90      print"[HOME]":poke2021,4:print"[SH
        IFT CLR][CRSR-DOWN]tape[SPACE]gere
        ed?[CRSR-UP]":return
100     rem-------------------------------
        --
110     rem  m e n u  2  ( m a t r i x )
120     rem-------------------------------
        --
130     mn=11:gosub520
140     gosub40
150     ift$="[CRSR-LEFT]"andx>0thenx=x-1
160     ift$="[CRSR-RIGHT]"andx<7thenx=x+1
170     ift$="[CRSR-UP]"andy>0theny=y-1
180     ift$="[CRSR-DOWN]"andy<7theny=y+1
190     ift$="+"thenc1$(x,y)="Q":gosub570
200     ift$="-"thenc1$(x,y)=".":gosub570
210     ift$="r"thengosub300
220     ift$="s"thengosub330
230     ift$="i"thengosub360
240     ift$="?"thengosub400
250     ift$="[SHIFT-CLR]"thengosub450
260     ift$=chr$(13)orj=>128thengosub480:
        goto630
270     goto140
280     :
290     rem-----roteer--------------------
        -
300     fory=0to7:forx=0to7:c2$(x,y)=c1$(7
        -y,x):next:next:gosub580:return
310     :
320     rem-----spiegel-------------------
        -
330     fory=0to7:forx=0to7:c2$(7-x,y)=c1$
        (x,y):next:next:gosub580:return
340     :
350     rem-----inverteer-----------------
        -
360     fory=0to7:forx=0to7:ifc1$(x,y)="Q"
        thenc1$(x,y)=".":elsec1$(x,y)="Q"
370     gosub570:next:next:x=0:y=0:return
380     :
```

```
390   rem-----random char---------------
      -
400   w=int(rnd(1)*5)+2:fory=0to7:forx=0
      to7:r=int(rnd(1)*w)
410   ifr=1thenc1$(x,y)="Q":elsec1$(x,y)
      ="."
420   gosub570:next:next:x=0:y=0:return
430   :
440   rem-----clear char----------------
      -
450   fory=0to7:forx=0to7:c1$(x,y)=".":n
      ext:char,13,7+y,w1$:next:x=0:y=0:r
      eturn
460   :
470   rem-----zet char in ram-----------
      -
480   forl=0to7:by=0:form=7to0step-1:ifc
      1$(7-m,l)="Q"thenby=byor(2[kwadraa
      tpijl]m)
490   next:pokebt+l,by:next:x=x1:y=y1:mc
      =cr:return
500   :
510   rem-----copieer char naar matrix--
      -
520   poke3293,cr:gosub450:bt=fnbyte(cr)
      :fory=0to7:k=y+bt:form=7to0step-1
530   x=7-m:if(peek(k)and(2[kwadraatpijl
      ]m))=0thenc1$(x,y)=".":elsec1$(x,y
      )="Q"
540   gosub570:next:next:x=0:y=0:return
550   :
560   rem-----subroutines---------------
      -
570   char,13+x,7+y,c1$(x,y):return
580   fory=0to7:forx=0to7:c1$(x,y)=c2$(x
      ,y):gosub570:next:next:x=0:y=0:ret
      urn
590   :
600   rem-------------------------------
      -
610   rem  m e n u 1 ( k i e s  c h a r
      )
620   rem-------------------------------
      -
630   mn=0
640   gosub40
650   ift$="[CRSR-LEFT]"andx>0thenx=x-1:
      cr=cr-1
660   ift$="[CRSR-RIGHT]"andx<7thenx=x+1
      :cr=cr+1
670   ift$="[CRSR-UP]"andy>0theny=y-1:cr
      =cr-8
680   ift$="[CRSR-DOWN]"andy<15theny=y+1
      :cr=cr+8
690   ift$="o"thengosub800
700   ift$="c"thengosub830
710   ift$="0"thengosub860
720   ift$="4"thengosub900
730   ift$="s"thengosub940
740   ift$="l"thengosub980
750   ift$=chr$(13)thenx1=x:y1=y:goto130
760   ift$=chr$(27)then1400
770   goto640
780   :
790   rem-----origineel karakter--------
      -
800   bt=fnbyte(cr):forl=0to7:pokebt+l,p
      eek(bt+l-1024):next:return
810   :
820   rem-----copieer matrix naar char--
      -
830   c2=cs+8*mc:bt=fnbyte(cr):forl=0to7
      :pokebt+l,peek(c2+l):next:return
840   :
850   rem-----achtergrondkleur----------
      -
860   ifac<16thenac=ac+1:elseac=1
870   color0,ac:return
880   :
890   rem-----borderkleur---------------
      -
900   ifbc<16thenbc=bc+1:elsebc=1
910   color4,bc:return
920   :
930   rem-----save charset naar tape----
      -
940   gosub90:gosub40:ift$<>"j"then1000:
      elseopen1,1,1,tp$:forl=cstocs+1023
950   print#1,peek(l);chr$(13);:next:pri
      nt#1:close1:goto1000
960   :
970   rem-----load charset van tape-----
      -
980   gosub90:gosub40:ift$<>"j"then1000:
      elseopen1,1,0,tp$:forl=cstocs+1023
990   input#1,vl:pokel,vl:next:close1
1000  gosub1350:poke2021,24:return
1010  :
1020  rem-------------------------------
      -
1030  rem  i n i t i a l i s a t i e
1040  rem-------------------------------
      -
1050  scnclr:color0,1:char,14,12,"[CTRL-
      2]even[SPACE]geduld[CTRL-1]":print
      "[HOME][6xCRSR-DOWN]t[SPACE]d000[S
      PACE]d7f5[SPACE]3800"
1060  print"[CRSR-DOWN]t[SPACE]d000[SPAC
      E]d400[SPACE]3400"
1070  print"[CRSR-DOWN]x":print"[CRSR-DO
      WN]run3020[HOME]":poke239,4:forl=1
      319to1322:pokel,13:next:monitor
1080  poke55,255:poke56,51:clr:v=65298:p
      okev,peek(v)and251:pokev+1,(peek(v
      +1)and3)or56
1090  vol8:color1,1:forl=1to38:l$=l$+"[C
      TRL 9][SPACE][CTRL-0]":l1$=l1$+"[C
      OM I]":next
1100  forl=1to8:keyl,"":next:cr=0:x=0:y=
      0:dimc1$(7,7),c2$(7,7):tp$="charse
      t"
1110  cs=14336:w$="[8xSPACE]":th=1021:mn
      =0:w1$="........":deffnbyte(cr)=14
      336+(8*cr)
1120  s1$="[SPACE]B"+w$+"B[SPACE]B"+w1$+
      "B[SPACE]B"+w$+"[6xSPACE]B":s4$="C
      CCCCCCC":s5$=s4$+"CCCCCC"
1130  s2$="[SPACE]B"+w$+"B[SPACE]B"+w$+w
      $+w$+"[SPACE]B":s6$="[COM-Q]"+s5$+
      "[COM-W]"
1140  :
1150  rem-------------------------------
      -
1160  rem  s c h e r m o p b o u w
1170  rem-------------------------------
      -
1180  gosub1350
1190  print"[3xSPACE]charset:[2xSPACE]ka
      rakter:[4xSPACE]commando's:"
1200  print"[SPACE][COM-A]"s4$"[COM-S][S
```

```
     PACE][COM-A]"s4$"[COM-S][SPACE][CO
     M A]"s5$"[COM-S]"
1210 data,,[+]-plot punt,[-]-wis punt,[
     r]-roteer,[s]-spiegel,[i]-invertee
     r
1220 data[?]-rnd char,[return]-vergroot
      char,[c]-copieer [o]-orig.char
1230 data[0] [4]-schermkleuren,[s]-save
      set [l]-load set,[esc]-basicloade
     r & stop
1240 forl=1to8:gosub1330:next
1250 print"[SPACE]B"w$"B[SPACE][COM-Z]"
     s4$"[COM-X][SPACE]B[shft/clr]-wisB
     "
1260 print"[SPACE]B"w$"B"w$"[4xSPACE]B[
     return]-terugB"
1270 print"[SPACE]B"w$"B[SPACE][COM-A]"
     s4$"CC[COM-E]"s5$"[COM-W]"
1280 f=11:forl=1to5:gosub1330:next
1290 print"[SPACE][COM-Z]"s4$"[COM-X][S
     PACE][COM-Z]"s5$;s4$"CCC[COM-X]";
1300 char,24,7,"[cursor]-move":char,23,
     8,s6$
1310 t=0:fork=0to15:forl=0to7:poke3072+
     (7+k)*40+l+2,t:t=t+1:next:next
1320 sound1,900,10:color0,2:goto630
1330 iff=0thens3$=s1$:elses3$=s2$
1340 readc$:mid$(s3$,25-f,len(c$))=c$:p
     rints3$:return
1350 print"[SHIFT-CLR][SPACE]"l1$"[2xSP
     ACE][CTRL-9][3xSPACE]commodore[SPA
     CE]16[2xSPACE]character[SPACE]desi
     gner[3xSPACE]":print"[SPACE]"l$
1360 print"[SPACE][CTRL-9][10xSPACE]ste
     ven[SPACE]bakker[SPACE]1987[10xSPA
     CE][CTRL-0][2xSPACE][CTRL-9]"l1$"[
     CTRL-0]";:return
1370 rem-------------------------------
     -
1380 rem  b a s i c l o a d e r
1390 rem-------------------------------
     -
1400 print"[HOME]":poke1341,0
1410 kb=1319:cr=peek(1341):bt=14336+8*c
     r:forl=0to1:form=0to7:c(l)=c(l)+pe
     ek(bt+m)
1420 next:bt=bt-1024:next:ifc(0)=c(1)th
     en1470
1430 rg$=rg$+str$(cr+5000)+"[SPACE]data
     "+str$(cr)+",":bt=14336+8*cr:forl=
     0to7
1440 s$=str$(peek(bt+l)):rg$=rg$+right$
     (s$,len(s$)-1)+",":next
1450 rg$=left$(rg$,len(rg$)-1):print"[S
     HIFT CLR]"rg$:print"run4060":poke2
     39,3:pokekb,19
1460 pokekb+1,13:pokekb+2,13:end
1470 poke1341,peek(1341)+1:ifpeek(1341)
     <128then1410
1480 print"[SHIFT-CLR]delete-4999":poke
     239,2:poke1319,19:poke1320,13:end
1490 data-1,-1,-1,-1,-1,-1,-1,-1,-1
1500 print"[SHIFT-CLR][6xCRSR-DOWN]t[SP
     ACE]d000[SPACE]d7f5[SPACE]3800":pr
     int"[CRSR-DOWN]x":print"[CRSR-DOWN
     ]run5202[HOME]":poke239,3
1510 forl=1319to1321:pokel,13:next:moni
     tor
1520 do:readsa:ifsa<0thenexit:elseforl=
     0to7:readbt:poke14336+8*sa+l,bt:ne
     xt:loop
1530 poke55,255:poke56,55:clr:v=65298:p
     okev,peek(v)and251:pokev+1,(peek(v
     +1)and3)or56
```

** EINDE LISTING cardes16 **

REGEL	10	79	REGEL	350	78	REGEL	690	20	REGEL	1030	88	REGEL	1370	47
REGEL	20	58	REGEL	360	230	REGEL	700	11	REGEL	1040	47	REGEL	1380	168
REGEL	30	12	REGEL	370	82	REGEL	710	251	REGEL	1050	49	REGEL	1390	47
REGEL	40	248	REGEL	380	58	REGEL	720	250	REGEL	1060	181	REGEL	1400	230
REGEL	50	244	REGEL	390	31	REGEL	730	29	REGEL	1070	154	REGEL	1410	137
REGEL	60	24	REGEL	400	46	REGEL	740	26	REGEL	1080	128	REGEL	1420	59
REGEL	70	58	REGEL	410	237	REGEL	750	145	REGEL	1090	147	REGEL	1430	207
REGEL	80	92	REGEL	420	82	REGEL	760	162	REGEL	1100	198	REGEL	1440	231
REGEL	90	15	REGEL	430	58	REGEL	770	35	REGEL	1110	249	REGEL	1450	29
REGEL	100	92	REGEL	440	242	REGEL	780	58	REGEL	1120	74	REGEL	1460	19
REGEL	110	28	REGEL	450	156	REGEL	790	248	REGEL	1130	150	REGEL	1470	159
REGEL	120	92	REGEL	460	58	REGEL	800	233	REGEL	1140	58	REGEL	1480	18
REGEL	130	13	REGEL	470	20	REGEL	810	58	REGEL	1150	47	REGEL	1490	49
REGEL	140	241	REGEL	480	236	REGEL	820	19	REGEL	1160	45	REGEL	1500	6
REGEL	150	99	REGEL	490	215	REGEL	830	107	REGEL	1170	47	REGEL	1510	85
REGEL	160	235	REGEL	500	58	REGEL	840	58	REGEL	1180	86	REGEL	1520	142
REGEL	170	90	REGEL	510	19	REGEL	850	19	REGEL	1190	178	REGEL	1530	130
REGEL	180	226	REGEL	520	181	REGEL	860	219	REGEL	1200	197			
REGEL	190	187	REGEL	530	113	REGEL	870	143	REGEL	1210	152			
REGEL	200	26	REGEL	540	82	REGEL	880	58	REGEL	1220	60			
REGEL	210	18	REGEL	550	58	REGEL	890	129	REGEL	1230	137			
REGEL	220	22	REGEL	560	163	REGEL	900	223	REGEL	1240	214			
REGEL	230	15	REGEL	570	146	REGEL	910	148	REGEL	1250	26			
REGEL	240	0	REGEL	580	108	REGEL	920	58	REGEL	1260	175			
REGEL	250	89	REGEL	590	58	REGEL	930	214	REGEL	1270	153			
REGEL	260	85	REGEL	600	47	REGEL	940	62	REGEL	1280	103			
REGEL	270	30	REGEL	610	144	REGEL	950	148	REGEL	1290	250			
REGEL	280	58	REGEL	620	47	REGEL	960	58	REGEL	1300	39			
REGEL	290	242	REGEL	630	125	REGEL	970	183	REGEL	1310	249			
REGEL	300	102	REGEL	640	241	REGEL	980	61	REGEL	1320	104			
REGEL	310	58	REGEL	650	85	REGEL	990	53	REGEL	1330	114			
REGEL	320	253	REGEL	660	220	REGEL	1000	70	REGEL	1340	118			
REGEL	330	102	REGEL	670	83	REGEL	1010	58	REGEL	1350	216			
REGEL	340	58	REGEL	680	9	REGEL	1020	47	REGEL	1360	107			

Gokken

Het volgende programma is er één voor de gokliefhebbers. Het is een programma dat we al een tijdje hebben liggen en waar geen inzender van bekend is.

```
10   rem ****************************
20   rem *gokken voor c-16 en plus/4*
30   rem ****************************
40   vol7
50   print"[SHIFT-CLR][CTRL-1]"
60   print"[3xSPACE]het[SPACE]spel[SPAC
     E]gokken[SPACE]met[SPACE]de[SPACE]
     computer"
70   print"[SPACE]wordt[SPACE]gespeeld[
     SPACE]met[SPACE]2[SPACE]'rollen'."
80   print"[SPACE]eerst[SPACE]moet[SPAC
     E]je[SPACE]je[SPACE]beginkapitaal"
90   print"[SPACE]opgeven.[SPACE]en[SPA
     CE]daarna[SPACE]wordt[SPACE]er[SPA
     CE]gevraagd"
100  print"[SPACE]hoeveel[SPACE]je[SPAC
     E]inzet.[SPACE]daarna[SPACE]probee
     r"
110  print"[SPACE]je[SPACE]het.[SPACE]a
     ls[SPACE]je[SPACE]een[SPACE]score[
     SPACE]van[SPACE]2,3"
120  print"[SPACE]of[SPACE]12[SPACE]heb
     t[SPACE]in[SPACE]het[SPACE]begin[S
     PACE]dan[SPACE]"
130  print"[SPACE]win[SPACE]je.[SPACE]a
     ls[SPACE]je[SPACE]een[SPACE]score"
140  print"[SPACE]van[SPACE]7[SPACE]of[
     SPACE]11[SPACE]hebt[SPACE]dan[SPAC
     E]verlies[SPACE]je."
150  print"[SPACE]als[SPACE]je[SPACE]ee
     n[SPACE]andere[SPACE]score[SPACE]d
     an"
160  print"[SPACE]2,3,7,11[SPACE]of[SPA
     CE]12[SPACE]hebt[SPACE]dan[SPACE]m
     oet[SPACE]je"
170  print"[SPACE]dat[SPACE]getal[SPACE
     ]wat[SPACE]je[SPACE]dan[SPACE]hebt
     [SPACE]halen."
180  print"[SPACE]als[SPACE]je[SPACE]da
     t[SPACE]getal[SPACE]dan[SPACE]hebt
     [SPACE]dan[SPACE]"
190  print"[SPACE]win[SPACE]je,maar[SPA
     CE]als[SPACE]je[SPACE]dan[SPACE]7[
     SPACE]of[SPACE]11"
200  print"[SPACE]hebt[SPACE]verlies[SP
     ACE]je.
210  print"[SPACE]je[SPACE]moet[SPACE]n
     et[SPACE]zolang[SPACE]doorgaan[SPA
     CE]"
220  print"[SPACE]totdat[SPACE]je[SPACE
     ]een[SPACE]van[SPACE]die[SPACE]dri
     e[SPACE](7,11
230  print"[SPACE]of[SPACE]het[SPACE]ge
     tal[SPACE]om[SPACE]te[SPACE]winnen
     )
240  print"[SPACE]haalt.
250  print
260  print"[SPACE]druk[SPACE]een[SPACE]
     toets[SPACE]om[SPACE]verder[SPACE]
     te[SPACE]gaan.
270  geta$
280  if a$=""then 270
290  rem *********************
300  rem *de 'rollen' opzetten*
310  rem *********************
320  a$="[9xSPACE]U[7xSHIFT-*]I[5xSPACE
     ]U[7xSHIFT-*]I"
330  b$="[9xSPACE][SHIFT--][SPACE][CRSR
     -RIGHT][5xSPACE][SHIFT--][5xSPACE]
     [SHIFT--][7xSPACE][SHIFT--]"
340  c$="[9xSPACE]J[7xSHIFT-*]K[5xSPACE
     ]J[7xSHIFT-*]K"
350  rem ************************
360  rem *het verkrijgen van een*
370  rem *begin kapitaal        *
380  rem ************************
390  print"[SHIFT-CLR]"
400  input"[3xSPACE]begin[SPACE]kapitaa
     l";c
410  rem ********************
420  rem *het starten van de *
430  rem *volgende weddeschap*
440  rem ********************
450  print"[3xSPACE]druk[SPACE]een[SPAC
     E]toets[SPACE]voor[21xSPACE]volgen
     de[SPACE]weddeschap"
460  get r$
470  if r$=""then 460
480  print"[3xSPACE]je[SPACE]kapitaal[S
     PACE]is[SPACE]nu";c
490  input"[3xSPACE]hoeveel[SPACE]zet[S
     PACE]je[SPACE]in";w
500  if w > 0then 530
510  print"[3xSPACE]doe[SPACE]niet[SPAC
     E]zo[SPACE]stom!"
520  goto 450
530  if w<=c then 590
540  print"[3xSPACE]zoveel[SPACE]heb[SP
     ACE]je[SPACE]niet!"
550  goto450
560  rem *************************
570  rem *organiseer het proberen*
580  rem *************************
590  print"[SHIFT-CLR][3xCRSR-DOWN][3xS
     PACE]eerste[SPACE]keer[SHIFT-SPACE
     ](inzet=";w;")"
600  print"[HOME][7xCRSR-DOWN]";a$
610  forj=1to5
620  printb$
630  nextj
640  printc$
650  rem ************************
660  rem *de cijfers verkrijgen*
670  rem ************************
680  q=int(10+50*rnd(0))
690  forz=1toq
700  a=int(1+6*rnd(0))
710  b=int(1+6*rnd(0))
720  rem ******************************
730  rem *het geluid bij het draaien *
740  rem *van de getallen voor de gok*
750  rem ******************************
760  sound1,700+3*(a*a+b*b),4
770  print"[HOME][10xCRSR-DOWN][12xCRSR
     -RIGHT]";a;"[11xCRSR-RIGHT]";b
780  next z
790  rem ******************************
800  rem *het optellen van de getallen*
810  rem ******************************
820  t=a+b
830  rem ******************
840  rem *overspringen als*
850  rem *de speler wint  *
860  rem ******************
```

```
870   if t=7 then 1360
880   if t=11 then 1360
890   rem *********************
900   rem *overspringen als de*
910   rem *speler verliest    *
920   rem *********************
930   if t=2 then 1600
940   if t=3 then 1600
950   if t=12 then 1600
960   print
970   print
980   print
990   print"[3xSPACE]je[SPACE]moet";t;"k
      rijgen[SPACE]voor[SPACE]7"
1000  print"[8xCRSR-DOWN][3xSPACE]druk[S
      PACE]een[SPACE]toets[SPACE]om[SPAC
      E]verder[SPACE]te[SPACE]gaan"
1010  get r$
1020  if r$ = "" then 1010
1030  print"[SHIFT-CLR][CRSR-DOWN][3xSPA
      CE]volgende[SPACE]kans(inzet=";w;"
      )"
1040  print"[3xSPACE]proberen[SPACE]om";
      t;"te[SPACE]krijgen"
1050  print"[HOME][4xCRSR-DOWN]"
1060  print a$
1070  forj=1to5
1080  print b$
1090  next j
1100  print c$
1110  rem ***********************
1120  rem *laat de getallen zien*
1130  rem ***********************
1140  q=int(10+10*rnd(0))
1150  for z=1 to q
1160  a=int(1+6*rnd(0))
1170  b=int(1+6*rnd(0))
1180  sound1,700+3*(a*a+b*b),4
1190  print"[HOME][8xCRSR-DOWN][12xCRSR-
      RIGHT]";a;"[11xCRSR-RIGHT]";b
1200  next z
1210  rem *************************
1220  rem *als a+b=t:despeler wint*
1230  rem *************************
1240  if a+b=t then 1360
1250  rem ***************************
1260  rem *als a+b=7:speler verliest*
1270  rem ***************************
1280  if a+b=7 then 1600
1290  rem ******************************
      **
1300  rem *anders doet de speler nog een
      s*
1310  rem ******************************
      **
1320  goto1000
1330  rem *************
1340  rem *speler wint*
1350  rem *************
1360  print"[7xCRSR-DOWN][3xSPACE]je[SPA
      CE]wint"
1370  rem ***************************
1380  rem *optellen van het gewonnen*
1390  rem *         bedrag          *
1400  rem ***************************
1410  c=c+w
1420  rem *********************
1430  rem *het winnende geluid*
1440  rem *********************
1450  forj=1to500:nextj
1460  sound1,834,32
1470  sound1,798,24
1480  sound1,810,8
1490  sound1,834,32
1500  sound1,739,32
1510  sound1,770,8
1520  sound1,798,8
1530  sound1,810,8
1540  sound1,834,8
1550  sound1,810,16
1560  sound1,798,16
1570  sound1,770,64
1580  goto450
1590  rem *****************
1600  rem *speler verliest*
1610  rem *****************
1620  print"[10xCRSR-DOWN][3xSPACE]je[SP
      ACE]verliest"
1630  rem ************************
1640  rem *het verliezende geluid*
1650  rem ************************
1660  for j=1to500:nextj
1670  for x=800to1000step 4
1680  sound1,x,1
1690  sound1,x+23,1
1700  nextx
1710  rem ***********************
1720  rem *het aftrekken van het*
1730  rem *  verloren bedrag    *
1740  rem ***********************
1750  c=c-w
1760  ifc=0then1780
1770  if c > 0then 450
1780  if ed= 2then 1810
1790  print"voor[SPACE]deze[SPACE]keer[S
      PACE]krijg[SPACE]je[SPACE]100[SPAC
      E]punten":c=100
1800  ed=2   :goto450
1810  print"je[SPACE]bent[SPACE]gebroken
      [SPACE]"
1820  fored=1to1000:next
1830  printchr$(147)"[SPACE]commodore[SP
      ACE]basic[SPACE]v3.5[SPACE]12277[S
      PACE]bytes[SPACE]free"
1840  end
```

** EINDE LISTING geldweg **.

REGEL	10	39	REGEL	260	154
REGEL	20	249	REGEL	270	6
REGEL	30	39	REGEL	280	38
REGEL	40	18	REGEL	290	43
REGEL	50	0	REGEL	300	255
REGEL	60	143	REGEL	310	43
REGEL	70	22	REGEL	320	23
REGEL	80	95	REGEL	330	237
REGEL	90	193	REGEL	340	7
REGEL	100	226	REGEL	350	127
REGEL	110	180	REGEL	360	120
REGEL	120	168	REGEL	370	143
REGEL	130	75	REGEL	380	127
REGEL	140	189	REGEL	390	112
REGEL	150	34	REGEL	400	243
REGEL	160	17	REGEL	410	1
REGEL	170	42	REGEL	420	83
REGEL	180	91	REGEL	430	15
REGEL	190	23	REGEL	440	1
REGEL	200	181	REGEL	450	236
REGEL	210	158	REGEL	460	23
REGEL	220	186	REGEL	470	56
REGEL	230	203	REGEL	480	112
REGEL	240	83	REGEL	490	124
REGEL	250	153	REGEL	500	2

REGEL 510	242	REGEL 780	220	REGEL 1050	52	REGEL 1320	74	REGEL 1590	89
REGEL 520	34	REGEL 790	123	REGEL 1060	254	REGEL 1330	177	REGEL 1600	28
REGEL 530	207	REGEL 800	225	REGEL 1070	135	REGEL 1340	240	REGEL 1610	89
REGEL 540	97	REGEL 810	123	REGEL 1080	255	REGEL 1350	177	REGEL 1620	132
REGEL 550	34	REGEL 820	51	REGEL 1090	204	REGEL 1360	37	REGEL 1630	127
REGEL 560	169	REGEL 830	131	REGEL 1100	0	REGEL 1370	253	REGEL 1640	187
REGEL 570	16	REGEL 840	101	REGEL 1110	85	REGEL 1380	109	REGEL 1650	127
REGEL 580	169	REGEL 850	121	REGEL 1120	16	REGEL 1390	136	REGEL 1660	237
REGEL 590	91	REGEL 860	131	REGEL 1130	85	REGEL 1400	253	REGEL 1670	101
REGEL 600	7	REGEL 870	57	REGEL 1140	93	REGEL 1410	57	REGEL 1680	236
REGEL 610	135	REGEL 880	100	REGEL 1150	179	REGEL 1420	1	REGEL 1690	251
REGEL 620	255	REGEL 890	1	REGEL 1160	242	REGEL 1430	214	REGEL 1700	218
REGEL 630	204	REGEL 900	238	REGEL 1170	243	REGEL 1440	1	REGEL 1710	85
REGEL 640	0	REGEL 910	28	REGEL 1180	16	REGEL 1450	237	REGEL 1720	37
REGEL 650	85	REGEL 920	1	REGEL 1190	139	REGEL 1460	103	REGEL 1730	245
REGEL 660	105	REGEL 930	49	REGEL 1200	220	REGEL 1470	113	REGEL 1740	85
REGEL 670	85	REGEL 940	50	REGEL 1210	169	REGEL 1480	52	REGEL 1750	58
REGEL 680	97	REGEL 950	98	REGEL 1220	210	REGEL 1490	103	REGEL 1760	39
REGEL 690	179	REGEL 960	153	REGEL 1230	169	REGEL 1500	107	REGEL 1770	239
REGEL 700	242	REGEL 970	153	REGEL 1240	47	REGEL 1510	57	REGEL 1780	105
REGEL 710	243	REGEL 980	153	REGEL 1250	253	REGEL 1520	67	REGEL 1790	163
REGEL 720	81	REGEL 990	54	REGEL 1260	88	REGEL 1530	52	REGEL 1800	201
REGEL 730	40	REGEL 1000	22	REGEL 1270	253	REGEL 1540	58	REGEL 1810	226
REGEL 740	77	REGEL 1010	23	REGEL 1280	15	REGEL 1550	99	REGEL 1820	14
REGEL 750	81	REGEL 1020	96	REGEL 1290	207	REGEL 1560	114	REGEL 1830	48
REGEL 760	16	REGEL 1030	43	REGEL 1300	47	REGEL 1570	107	REGEL 1840	128
REGEL 770	173	REGEL 1040	135	REGEL 1310	207	REGEL 1580	34		

Kubus c 16

M Podda uit Enschede is vast een actief kubus puzzelaar, hij stuurde ons namelijk de computerversie van dit zenuwslopende spel op. Een tweedimensionale uitvoering waarbij de rijen naar rechts of naar beneden schuiven d.m.v. de cijfertoetsen.

```
10   rem *******************************
     ***
20   rem *****    kubus    c-16       **
     ***
30   rem *****  bewerkt door m.podda **
     ***
40   rem *****     origineel door     **
     ***
50   rem *****        e.de ruiter     **
     ***
60   rem *******************************
     ***
70   print"[SHIFT-CLR]":color0,1:color4
     ,1:color1,5
80   dima%(5,5):fori=0to4:reada%:forj=0
     to4:a%(i,j)=a%:nextj,i
90   data28,5,31,158,30
100  goto260
110  rem ** scherm **
120  print"[SHIFT-CLR][10xCRSR-RIGHT]UC
     6CC7CC8CC9CC0CI"
130  fori=5to1step-1:printspc(10);"B";s
     pc(15);"B"
140  a$=right$(str$(i),1):printspc(9);a
     $;"B";spc(15);"B";a$
150  printspc(10);"B";spc(15);"B"
160  next :print"[3xSPACE][7xCRSR-RIGHT
     ]JC6CC7CC8CC9CC0CK":print"[2xCRSR-
     DOWN][CTRL-9]o[CTRL-0]pnieuw":prin
     t"[CRSR-DOWN][CTRL-9]s[CTRL-0]topp
     en"
170  print"[HOME]";:fori=0to4:forj=1to3
     :print:printspc(10)"[CTRL-9][CRSR-
     RIGHT]";:fork=0to4
180  printchr$(a%(i,k));"[3xSPACE]";:ne
     xtk,j,i
190  return
200  rem ** verdraaien **
210  fori=5to1step-1:a%(a,i)=a%(a,i-1):
     next
220  a%(a,0)=a%(a,5):return
230  fori=5to1step-1:a%(i,a)=a%(i-1,a):
     next
240  a%(0,a)=a%(5,a):return
250  rem ** hoofdprogramma **
260  gosub120:forw=0to20:a=int(5*rnd(1)
     ):on1+2*rnd(1)gosub210,230:gosub17
     0:nextw
270  geta$:ifa$="s"thengosub340:gosub34
     0:gosub340:sys 65529
280  gosub450:a=val(a$):ifa$="0"thena=1
     0:
290  ifa=0then300
300  ifa>5thena=a-6:gosub230:goto320
310  a=5-a:gosub210
320  gosub170:goto270
330  rem ** programma ii **
340  print"[SHIFT-CLR]":color0,1:color4
     ,1
350  a=3572:k=2548:s=81:pokea,s
360  forb=1to10:pokea+b,s:pokea-b,s
370  pokea+(40*b),s:pokea-(40*b),s
380  pokea+(40*b+b),s:pokea-(40*b-b),s
390  pokea-(40*b+b),s:pokea+(40*b-b),s:
     nextb
400  forkc=1to15:forb=1to10:pokek,kc
410  pokek+b,kc:pokek-b,kc
420  pokek+(40*b),kc:pokek-(40*b),kc
430  pokek+(40*b+b),kc:pokek-(40*b-b),k
     c
440  pokek-(40*b+b),kc:pokek+(40*b-b),k
     c:nextb:nextkc:return
450  ifa$="o"thenrestore:gosub340:run70
460  return
470  rem ** einde listing **
```

** EINDE LISTING kubus-16 **

Checksum-getallen op de volgende pagina.

REGEL 10	249	REGEL 250	77
REGEL 20	148	REGEL 260	127
REGEL 30	94	REGEL 270	227
REGEL 40	5	REGEL 280	83
REGEL 50	10	REGEL 290	232
REGEL 60	249	REGEL 300	34
REGEL 70	131	REGEL 310	110
REGEL 80	186	REGEL 320	129
REGEL 90	55	REGEL 330	111
REGEL 100	33	REGEL 340	208
REGEL 110	249	REGEL 350	7
REGEL 120	220	REGEL 360	166
REGEL 130	0	REGEL 370	131
REGEL 140	249	REGEL 380	92
REGEL 150	187	REGEL 390	90
REGEL 160	158	REGEL 400	183
REGEL 170	236	REGEL 410	75
REGEL 180	212	REGEL 420	13
REGEL 190	142	REGEL 430	230
REGEL 200	24	REGEL 440	246
REGEL 210	105	REGEL 450	241
REGEL 220	39	REGEL 460	142
REGEL 230	105	REGEL 470	180
REGEL 240	39		

Tape Disk

Richard van Gelder uit Koedijk stuurde ons het volgende programma. Het is een in Basic geschreven back-up programma. Hiermee wordt de inhoud van een tape naar disk gekopieerd. Vanuit het programma wordt de nodige informatie en instructies gegeven.

```
10   rem **********************************
20   rem *                                 *
30   rem * kopieerhulp van tape naar disk *
40   rem * voor c-16 & plus/4             *
50   rem * door richard van gelder        *
60   rem * koedijk                         *
70   rem *                                 *
80   rem **********************************
90   trap330:color0,2:color4,2,4:x$=chr$(34)
100  ifpeek(806)=63theny$="quit:":z$="sY1760:":elsey$="end:":z$=""
110  print"[SHIFT-CLR][CTRL-9][40xSPACE]"
120  print"[CTRL-9][CTRL-1]***[2xSPACE]kopieerhulp[SPACE]van[SPACE]tape[SPACE]naar[SPACE]disk[2xSPACE]***"
130  print"[CTRL-9][40xSPACE]"
140  print"[5xCRSR-DOWN]een[SPACE]programma[SPACE]voor[SPACE]het[SPACE]"x$"overzetten"x$"[SPACE]van"
150  print"[CRSR-DOWN](alle)[SPACE]programma's[SPACE]van[2xSPACE]tape[SPACE]naar[2xSPACE]disk.
160  print"[5xCRSR-DOWN][CTRL-9]1[CTRL 0][SPACE]=[SPACE]programma[2xSPACE][CTRL-3]stoppen[CTRL-1][2xSPACE]na[SPACE]een[SPACE]laadfout!"
170  print"[CRSR-DOWN][CTRL-9]2[CTRL-0][SPACE]=[SPACE]programma[SPACE][CTRL-3]vervolgen[CTRL-1][SPACE]na[SPACE]een[SPACE]laadfout!"
180  print"[CRSR-DOWN][13xSPACE][CTRL-4]maak[SPACE]een[SPACE]keuze[CTRL-1]"
190  getkeya$
200  ifa$="1"thenva$="?"+x$+"[2xCRSR-UP]helaas!![CTRL-2][CRSR-UP]"+x$+":vO8:sO1,50,120:cL":goto220
210  ifa$="2"thenva$="?"+x$+"[CTRL-2][3xCRSR-UP][12xSPACE][CRSR-UP]"+x$+":pO1319,19:pO239,10":else190
220  print"[5xCRSR-UP]zet[SPACE]de[SPACE]"x$"randapparatuur"x$"[SPACE]klaar[SPACE]en[SPACE]druk[SPACE]op"
230  print"[CRSR-DOWN]de[SPACE][CTRL-9]spatiebalk.[CTRL-0][SPACE](indien[SPACE]een[SPACE]printer[SPACE]klaar"
240  print"[CRSR-DOWN]staat,[SPACE]wordt[SPACE]een[SPACE]inhoudsopgave[SPACE]geprint)."
250  getkeya$:ifpeek(2038)<>60then250
260  print"[SHIFT-CLR][CTRL-2]oP4,4:cM4:pR4:clO4:gO230":print"[3xCRSR-DOWN]clO4:gO240"
270  print"[CTRL-1][2xCRSR-DOWN][14xSPACE]checking!!!!"
280  print"[CTRL-2]":directory"x"
290  ifds=74thenprint"[SHIFT-CLR][2xCRSR-DOWN][CTRL-1]geen[SPACE](juiste)[SPACE]disk[SPACE]in[SPACE]de[SPACE]drive[SPACE]aanwezig!":gosub360:goto250
300  poke1319,19:fora=0to10:poke1320+a,13:nexta:poke239,10:end
310  b$="lO"+x$+"$"+x$+",8":c$="oP4,4:cM4:lI:pR4:clO4:coL1,1":goto370
320  b$="?"+x$+"[CRSR-DOWN]":c$="vO8:sO1,800,200:coL1,1":goto370
330  ifer=30thenscnclr:end
340  ifer=5thenprint"[SHIFT-CLR][CTRL-1][5xCRSR-DOWN][7xSPACE]diskdrive[SPACE]niet[SPACE]gereed!":gosub360:resume250
350  resume
360  print"[2xCRSR-DOWN][SPACE]maak[SPACE]in[SPACE]orde[SPACE]en[SPACE]druk[SPACE]op[SPACE]de[SPACE]spatiebalk!":return
370  ifpeek(2044)<>4then400
380  print"[SHIFT-CLR][CTRL-1][7xCRSR-DOWN]druk[SPACE]op[SPACE]de[SPACE]'play'[SPACE]toets[SPACE]van[SPACE]de[SPACE]datasette"
390  ifpeek(1)<>192then390:elsecolor0,2
400  rem **opzetten beeldschermgeheugen**
410  print"[CTRL-2][SHIFT-CLR]coL4,2,4:?tA49)"x$"[CTRL-1][SPACE]on[SPACE]tape...[2xSPACE][CTRL-2]"x$;
420  print":?"x$"[CRSR-DOWN][4xSPACE]scR(f$):end:"x$":?"x$"[CRSR-DOWN]"y$x$":?"x$"[CTRL-1][HOME]"x$":"z$"lO"
430  printtab(13)"y"tab(165)x$"[6xSPACE][CTRL-1]"x$"d$"x$"[CTRL-2][4xCRSR-UP]"x$":coL4,2,4:"b$
440  printx$"[13xSPACE]saving[SPACE]on[SPACE]disk[SPACE]:"x$":?"x$"[CTRL-2][5xCRSR-DOWN]
```

```
450   printva$
460   print"i$="x$"[2xSPACE][CTRL-1]spoe
      l[SPACE]de[SPACE]tape[SPACE]terug[
      SPACE]tot[SPACE]de"x$":?"x$"[2xCRS
      R-UP]"
470   print"x$=cH(34):?"x$"[CTRL-2][HOME
      ][4xCRSR-DOWN][2xSPACE]f$="x$"x$:?
      "x$"[10xSPACE]"x$;
480   print":?"x$"[CRSR-DOWN]?"x$"x$"x$"
      [5xCRSR-UP][CTRL-1]"x$":?"x$"[CRSR
      -DOWN]i$="x$"x$"x$"[2xSPACE]"x$":?
      "x$"[HOME][CRSR-DOWN]"x$
490   print"r$="x$"[2xSPACE]en[SPACE]laa
      d[SPACE]kopieerhulp[SPACE]opnieuw.
      "x$":?"x$"[2xCRSR-UP]"
500   print"iff$<>"x$x$"thendS(f$):else"
      ;
510   print"f$="x$"naamloos"x$":x=1";
520   print"[4xSPACE]:?"x$"[2xCRSR-UP]+"
      x$"x";
530   print":f$=f$+stR(x):?"x$"[HOME][CT
      RL 9][CTRL-7]"x$"tA206)f$"x$"[CTR
      L 2][4xCRSR-DOWN]"
540   print"?"x$"[6xCRSR-UP]"x$"i$"x$"[S
      PACE]'misser!'"x$",r$"x$"[CTRL
       2]"x$":?"x$"[4xCRSR-DOWN]"
550   print"[CRSR-DOWN][6xSPACE][COM-4][
      CTRL-9]*tape*[CTRL-0][COM-8][2xS
      PACE]'kopieerhulp'[2xSPACE][COM-4]
      [CTRL-9]*disk*[CTRL-0][CTRL-2]"
560   print"?"x$"[9xCRSR-UP]r$="x$"x$"x$
      "[2xSPACE]"x$":?:ifds<20ords=63the
      n";
570   print"p01319,19:p0239,10:new:else?
      tA202)ds"x$"[CRSR-UP]"
580   print"if[2xSPACE]0=72then"c$"
590   print"[COM-8][7xSPACE]richard[SPAC
      E]van[SPACE]gelder,[SPACE]koedijk[
      CTRL-2]"
600   print"d$=mI(ds$,4,(len(ds$)-9)):?"
      x$"[CTRL-1][HOME][2xCRSR-DOWN][2xS
      PACE]scratch[CTRL-2]"x$"tA120)";
610   printx$"?"x$"x$"x$"[CRSR-DOWN][3xS
      PACE]"x$":p01319,19:p0239,10";
620   print"[HOME]":poke1319,19:fork=0to
      10:poke1320+k,13:next:poke239,10:n
      ew
```

** EINDE LISTING tapedisk **

REGEL 10	35	REGEL 320	70
REGEL 20	227	REGEL 330	128
REGEL 30	135	REGEL 340	174
REGEL 40	205	REGEL 350	214
REGEL 50	172	REGEL 360	148
REGEL 60	27	REGEL 370	59
REGEL 70	227	REGEL 380	245
REGEL 80	35	REGEL 390	150
REGEL 90	22	REGEL 400	22
REGEL 100	17	REGEL 410	70
REGEL 110	130	REGEL 420	143
REGEL 120	31	REGEL 430	57
REGEL 130	239	REGEL 440	84
REGEL 140	82	REGEL 450	84
REGEL 150	229	REGEL 460	47
REGEL 160	157	REGEL 470	193
REGEL 170	233	REGEL 480	16
REGEL 180	147	REGEL 490	173
REGEL 190	255	REGEL 500	43
REGEL 200	152	REGEL 510	153
REGEL 210	10	REGEL 520	182
REGEL 220	171	REGEL 530	125
REGEL 230	117	REGEL 540	225
REGEL 240	75	REGEL 550	163
REGEL 250	172	REGEL 560	35
REGEL 260	169	REGEL 570	100
REGEL 270	79	REGEL 580	250
REGEL 280	166	REGEL 590	63
REGEL 290	66	REGEL 600	169
REGEL 300	163	REGEL 610	225
REGEL 310	29	REGEL 620	194

prijsvraag

De inzendtermijn voor de Commodore Info prijsvraag loopt nog tot 15 februari 1988. Stuur dus uw Amiga, C-16, C-64 of C-128 programma zo snel mogelijk in!

Voorwaarden:

* Maximale lengte van het programma: 150 regels. Inzendingen voor serieuze toepassingen mogen wat langer zijn.
* Alleen originele programma's mogen worden ingezonden, de inzender blijft hiervoor verantwoordelijk. Inzending alleen op magnetisch medium (cassette of diskette).

Categoriën

1) Commodore C-16, C-64, C-128, Amiga
2) Hardware-aanpassingen

Wat is er te winnen?

* Een kleurenmonitor
* Een monochrome monitor
* Een diskdrive
* Een groot aantal prachtige softwarepakketten.

N.B. Door inzending stemt men toe in publikatie, ook in magnetische vorm. De vergoeding bij plaatsing wordt door de redaktie bepaald. Inzenders krijgen na ontvangst van hun inzending een lege cassette of diskette toegestuurd, hou dus zelf een kopie van het programma.

Inzendingen:

Postbus 112, 1260 AC Blaricum,
o.v.v. **'Prijsvraag Commodore'**

Iedere 10e inzender ontvangt een verrassing!

Sluitingsdatum: 15 februari 1988

Titel	Prijs	Omschrijving	Leverancier
	ƒ 130,--	Utility	
HardHat	ƒ 170,--	Utility	*WestCom Inc*
HexDump	ƒ 50,--	Hex dump utilities	*Northwest Machine*
I.C.E	ƒ 200,--	Calculator plotter	*Quicksilver Software*
ICONizer	ƒ 75,--	Ontwerpen van Icons	*Royal Software*
Interchange	ƒ 520,--	Converteren van objecten	*Syndesis*
Jet Set	ƒ 100,--	HP LaserJet plus driver	*C Ltd*
Jet Set Font Sets	ƒ 125,--	Fonts voor JetSet	*C Ltd.*
Keep Track	ƒ 280,--	Utility	*The Other Guys*
Key Genie	ƒ 130,--	Utility	*Diskcovery*
KickWork	ƒ 100,--	Kickstart/Workbench op een disk	*Amigo Business Computers*
LTR	ƒ 150,--	Print utilities	*MetaSoft Limited*
Laser Fonts I	ƒ 100,--	Fonts voor LaserWriter	*S Anthony Studios*
Laser Utilities I	ƒ 100,--	IFF Files naar Postscript	*S Anthony Studios*
LaserUp!	ƒ 200,--	Print het scherm naar LaserWriter	*S Anthony Studios*
LaserUp!Plot	ƒ 120,--	Aegis tekeningen naar PostScript	*S Anthony Studios*
MagiCode	ƒ 70,--	Vertaalt computer data	*Magic Circle*
Marauder II	ƒ 100,--	Kopieer programma	*Discovery Software*
Math-Amation	ƒ 200,--	Reken utilities	*Taurus-Impex*
MergeMaster	ƒ 60,--	Mail-merge programma	*Megatronics*
MicroShell	ƒ 200,--	AmigaDOS Interface	*MetaSoft Limited*
Mirror Hacker Package, the	ƒ 120,--	Disk Analyseren	*Compumed*
Mirror, the	ƒ 125,--	Kopieerdisk	*Compumed*
Newsletter Fonts Vol I	ƒ 80,--	120 verschillende AMIGA fonts	*Inter/Active Software*
Outline!	ƒ 120,--	Utilities	*PAR Software*
PED	ƒ 320,--	Programeer tekstverwerker	*MicroDimensions*
Printer Drivers	ƒ 100,--	Diverse printer drivers	*Tychon Technologies*
Printer tekeningen	ƒ 18,--	Stuurt tekeningen naar de printer	*Courbois Software*
Professional Text Engine	ƒ 200,--	Teksteditor met macro's	*Zirkonics Corp*
Programma bestand	ƒ 18,--	Opslaan van een programma bestand	*Courbois Software*
Quarterback	ƒ 170,--	Harddisk naar diskette backup	*Central Coast Software*
Quick Nibble	ƒ 130,--	Kopieerprogramma	*Copperstate*
Shell	ƒ 189,--	AmigaDos interface	*Metacomco*
SmartDisk			*Know Technology*
	ƒ 150,--	Utility	
Software Enhancer	ƒ 37,--	DOS 1.2 voor de Amiga 1000	*Commodore*
Soundshop	ƒ 125,--	Kreëren van speciale muziektonen	*Revolution Software*
Studio Fonts Vol I	ƒ 95,--	Kleurenfonts	*Inter/Active Software*
Super Parameters	ƒ 80,--	Kopieerutilities	*Utilities Unlimited*
System Monitor	ƒ 125,--	Monitor geheugen en utilities	*Zen Software*
Transformer	ƒ 200,--	Vertaalprogramma voor MS DOS	*Commodore*
TxEd	ƒ 100,--	Programma tekst editor	*Microsmiths*
Tychon Utilities	ƒ 525,--	Disk utilities	*Tychon Technologies*
Viditel simulatie	ƒ 18,--	Bestands programma als viditel	*Courbois Software*
ZING!	ƒ 279,--	AmigaDos van de workbench	*Meridian Software Inc*
ZING!Keys	ƒ 189,--	Macro utilities	*Meridian Software Inc*
Zuma Fonts I	ƒ 85,--	Diverse Fonts	*Brown Waugh*
Zuma Fonts II	ƒ 85,--	Diverse Fonts	*Brown Waugh*
Zuma Fonts III	ƒ 85,--	Diverse Fonts	*Brown Waugh*

Zakelijk

Titel	Prijs	Omschrijving	Leverancier
ABase	ƒ 280,--	Database	*Computer Power*
AMT	ƒ 120,--	Kosten berekening	*The Other Guys*
Accounting	ƒ 250,--	Boekhoudprogramma	*ComputerWare*
Amiga Scientific Calculator	ƒ 50,--	Uitgebreide rekenmachine	*Deskware*
Analyze!	ƒ 370,--	Grafisch	*Micro Systems*
Analyze 1.0	ƒ 450,--	Berekeningen maken	*Micro systems*
Arima Techniques	ƒ 240,--	Time-series van Box Jenkins	*LionHeart Press Inc*
BEST Basic Ledger	ƒ 200,--	Grootboek	*B.E.S.T. Inc*
BEST Business Management 2.0	ƒ 980,--	Boekhoudprogramma	*B.E.S.T. Inc*
Business Static	ƒ 350,--	Statistiek en boekhouding	*LionHeart Press Inc*
CCI Bottom Liner	ƒ 500,--	Boekhoudprogramma	*ClockWork Computers Inc*
CCI Integrated Merchandiser	ƒ 1250,--	Boekhouding en voorraadadministratie	*ClockWork Computers Inc*
Cluster Analysis	ƒ 280,--	Cluster analyses	*LionHeart Press Inc*
Decision Analysis Techniques	ƒ 340,--	Boekhoudprogramma	*LionHeart Press Inc*
Econometrics	ƒ 360,--	Economische berekeningen	*LionHeart Press Inc*
Experimental Statistic	ƒ 360,--	Uitvoeren van statistische bewerkingen	*LionHeart Press Inc*
Exploratory Data Analysis			*LionHeart Press Inc*

Uit het grote aantal inzendingen blijkt, dat de Basic Micro's zich mogen verheugen in een behoorlijke populariteit, en dat is voor ons reden genoeg om ook in het nieuwe jaar met deze rubriek door te gaan. Ons adres is: Postbus 112, 1260 AC, Blaricum.

Basic Micro's



Deze twee pagina's worden onderhand voor de helft gevuld door een vast aantal "medewerkers" in het land. Het lijkt ons echter wel leuk om ook de programma's van andere talenten te ontvangen, zodat er wat meer variatie in de namen ontstaat. Vooral van de dames ontvangen we nauwelijks iets. Nou is het vrij bekend dat overwegend de heren in computers geïnteresseerd zijn, maar er zullen toch wel een paar digitale dames rondlopen?

Beeldstoring

Wij ontvingen van Obbe Vermeij uit Capelle aan de IJssel weer eens twee brieven met enkele Micro's. We hebben er twee uitgekozen. De eerste veroorzaakt een "storing" in het beeld, maar dat lijkt ons geen reden om de reparateur te bellen. Probeer eens het tweede getal 48 in regel 30 te veranderen, en vergeet dan de checksum niet.

```
10 FORT=0TO82:READP:C=C+P:POKE49152+T,P:
NEXT:IFC=8232THENSYS49153:LIST
20 DATA112,120,169,19,141,20,3,169,192,1
41,21,3,169,129,141,26,208,88,96,173
30 DATA25,208,141,25,208,48,3,76,49,234,
160,15,162,48,202,208,253,173,22,208
40 DATA41,240,25,68,192,141,22,208,136,1
6,237,238,0,192,173,0,192,141,18
50 DATA208,169,27,141,17,208,76,188,254,
0,1,2,3,4,5,6,7,7,6,5,4,3,2,1,0
```

De tweede is een antwoord op de oproep voor niet-grafische Micro's, het getal e (2.7182818284590452353...) wordt er namelijk door benaderd. Dat kan met: e = 1/0! + 1/1! + 1/2! + 1/3! + 1/4! + 1/5! + ...

```
10 E=1:Y=1:FORT=1TO15:Y=Y*T:E=E+1/Y:PRIN
TE:NEXT
```

Smooth-scroll-regel

Als eigen bijdrage dit keer een letterlijk flitsende smooth-scroll demo voor de C-64, zonder geheimzinnige machinetaal. Tik maar in en verwijder vervolgens willekeurig een van de drie REM-opdrachten die vooraan de regels 50 tot en met 70 staan. Vergeet dit niet, anders krijg je een '?syntax error in 80'. Deze regels bepalen de kleuren die je krijgt.

```
10 DIMC$(7):FORT=0TO7:READC:C$(T)=CHR$(1
9)+CHR$(C):NEXT:FORT=0TO2:READA$
20 B$=B$+A$:NEXT:FORT=1TO13:B$="    "+B$+
"    ":NEXT:POKE53280,0:POKE53281,0
30 PRINTCHR$(147):FORA=1TOLEN(B$)-39:N$=
MID$(B$,A,40):FORT=0TO7:PRINTC$(T)N$
40 POKE52370,199-T:NEXTT,A:GOTO30
50 REMDATA144,151,152,155,5,155,152,151
60 REMDATA144,28,129,158,5,158,129,28
70 REMDATA144,31,154,159,5,159,154,31
80 DATA"DIT IS EEN LEUKE SMOOTH-SCROLL-R
EGEL, DIE MET SIMPELE "
90 DATA"MIDDELEN IS TE MAKEN. HET IS NOO
DZAKELIJK DAT DE EERSTE "
100 DATA"KLEUR HETZELFDE IS ALS DIE VAN
HET SCHERM, ANDERS VERSPRINGT ALLES."
```

Je kunt ook nog een regel tussen 20 en 30 toevoegen die eerst het hele scherm vult met hetzelfde karakter, zodat alles naar links gaat 'scrollen'. Verwijder dan wel de PRINTCHR$(147) in regel 30.

Amiga

We werden door David Vanhove uit Maaseik-Neeroeteren (België) geattendeerd op een fout in de vorige uitgave in een listing voor de Amiga, die alweer te wijten is aan onzorgvuldig overtikken. In de eerste Micro moet de tweede regel er als volgt uitzien:

```
WINDOW 2,"Cirkels",(160,0)-(320,100),0,1
```

Stuurden Ron en Kim Brobbel uit Vlaardingen de vorige keer nog een Micro in voor de C-64, nu doen ze het voor de Amiga. Het is een klein tekenprogramma in Amiga-Basic. De besturing gaat met de muis en de linker muisknop. Eerst moet je een kleur kiezen, daarna kun je links gaan tekenen, en alles verschijnt rechts in het klein. Probeer ook eens de waardes van D en G te veranderen.
Denk eraan dat je je niets moet aantrekken van de hier gebruikte regellengte van 40 karakters!

```
SCREEN 1,640,512,4,4:WINDOW 1,"RAK SOFTW
ARE 1987",,0,1:G=5:D=525
FOR X=0 TO D STEP G:LINE (X,0)-(X,500),8
:NEXT
FOR Y=0 TO 500 STEP G:LINE (0,Y)-(D,Y),8
:NEXT
FOR Y=0 TO 15:LINE (D+1,Y*20)-(632,Y*20+
20),Y,BF:NEXT
PALETTE 0,0,0,0:PALETTE 1,1,.9,.3
RAK:
S=MOUSE(0):X=INT(MOUSE(1)/G):Y=INT(MOUSE
(2)/G):IF S=0 THEN RAK
IF X>INT(D/G-1) AND Y<INT(320/G) THEN GO
```

```
SUB KLEUR
IF X>INT(D/G-1) THEN RAK
PRESET (D+1+X,340+Y),K
X=X*G:Y=Y*G:LINE (X+1,Y+1)-(X+G-1,Y+G-1)
,K,BF:GOTO RAK
KLEUR:
K=INT(MOUSE(2)/10):LINE (D+1,460)-(632,5
00),K,BF:RETURN
```

Karakterset

Het verplaatsen van de karakterset van de C-64 is met Basic een langdurig proces. Bovendien is het voor gebruikers van de Power Cartridge niet mogelijk om dit te doen zonder vooraf QUIT in te tikken of het ding er gewoon uit te trekken. Machinetaal biedt hier de oplossing. Coen Antens uit Rucphen schreef een routine die de karakterset snel en probleemloos kopiëert naar de adressen vanaf 8192. Je kunt de routine in vrijwel elk programma opnemen.

```
10 FORI=0TO47:READQ:POKE49152+I,Q:C=C+Q:
NEXT:IFC=6026THENSYS49152:PRINT"OK"
20 DATA169,208,141,21,192,169,32,141,24,
192,120,169,51,133,1,162,16,160,0,185,0
30 DATA224,153,0,48,200,208,247,238,21,1
92,238,24,192,202,208,236,169,55,133,1
40 DATA88,169,25,141,24,208,96
```

Bij wijze van voorbeeld volgt hier hoe het sterretje uit de eerste karakterset veranderd kan worden in een copyright-teken.

```
50 FORI=0TO7:READQ:POKE8192+42*8+I,Q:NEX
T:LIST:DATA60,66,153,161,161,153,66,60
```

Algemeen geldt dat de 8 getallen die een karakter definiëren volgen op een adres dat als volgt bepaald wordt: adres = beginwaarde + schermcode * 8. De beginwaarde is hier dus 8192 voor de eerste karakterset, en 10240 voor de tweede. De schermcodes zijn te vinden in de diverse handboeken.

Naamtoren

Van Mark Weyn uit Nieuw-Namen ontvingen we een briefje met een Micro genaamd Naamtoren, die een woord of zin pyramide-vormig afdrukt. Het programma was hier en daar onnodig ingewikkeld, en hier staat onze aangepaste versie.
Een heel ander effect krijg je als je tussen PRINT en TAB in regel 20 ook nog CHR$(19) zet.

```
10 PRINTCHR$(147):INPUT"NAAM OF ZIN";A$:
X=LEN(A$)/2:B=INT(X+.5):PRINTCHR$(147)
20 C=2*(INT(X)-X):FORI=0TOB-1:PRINTTAB(B
-I-1)MID$(A$,B-I,2*I+2+C):NEXT
```

Wie dit programma wil aanpassen voor een IBM-compatibele met GW-Basic moet de TAB(B-I-1) veranderen in TAB(B-I), omdat met GW-Basic TAB(0) hetzelfde effect heeft als TAB(1).

Ellipsen

Patrick van Oostrom uit Utrecht stuurde een Micro voor de C-128 in die een uit ellipsen opgebouwde cirkel tekent.

```
10 COLOR0,1:COLOR4,1:GRAPHIC1,1:FORA=2TO
16:COLOR1,A:FORI=0TO500STEP25
20 CIRCLE1,140,100,100,10,I,I,I:NEXTI,A:
SLEEP1:GRAPHIC0,1
```

Box-opdracht

Eveneens voor de C-128 zijn enkele Micro's die we nog hadden liggen van Bernard Hudepohl uit Lelystad. Hij maakt gebruik van de veelzijdige opdracht BOX.

```
10 COLOR0,1:COLOR4,1:GRAPHIC1,1:FORX=0TO
110STEP3:BOX1,90,X,210,110:NEXT
20 FORY=0TO360STEP3:BOX1,90,X,210,110,Y:
NEXT
```

```
10 COLOR0,1:COLOR4,1:GRAPHIC1,1:FORX=0TO
180STEP3:IFX=90THENSLEEP3
20 BOX1,90,90,210,110,X:NEXT
```

Motor Racing met de C-16

Achterop een briefkaart kan het ook, moet R. Vletter uit Vlaardingen gedacht hebben. Motor Racing is de naam van deze Micro, die vooral geluid produceert.

```
10 SCNCLR:COLOR0,1:COLOR4,1:COLOR1,2:VOL
5:T=90
20 J=JOY(1):SOUND3,T,5:IFJ=1THENT=T+9:IF
T>891THENT=891
30 IFJ=5THENT=T-9:IFT<90THENT=90
40 PRINTTAB(14)CHR$(19)T/9" KM/H":GOTO20
```

Diskette-naam

Met het volgende programma voor de C-64, ingestuurd door Michel Fidder uit Barendrecht, kun je de naam van een diskette veranderen.

```
10 CLR:POKE53280,0:POKE53281,0:PRINTCHR$
(147):PRINT"DISK NAME CHANGE":PRINT
20 OPEN15,8,15,"I":OPEN1,8,2,"#":PRINT#1
5,"U1:";2;0;18;0:PRINT#15,"B-P";2;144
30 A$="                ":IFA=0THENINPUT"
TYPE THE NEW NAME";B$
40 B$=LEFT$(B$+A$,16):FORI=1TO16:PRINT#1
,MID$(B$,I,1);:NEXT
50 PRINT#15,"U2:";2;0;18;0:CLOSE15:CLOSE
1:IFA=1THENRETURN
60 A=1:GOSUB20:INPUT"ONE MORE TIME";I$:I
FI$="Y"THENRUN
```

Peter Broekhuizen

Amiga Nieuws

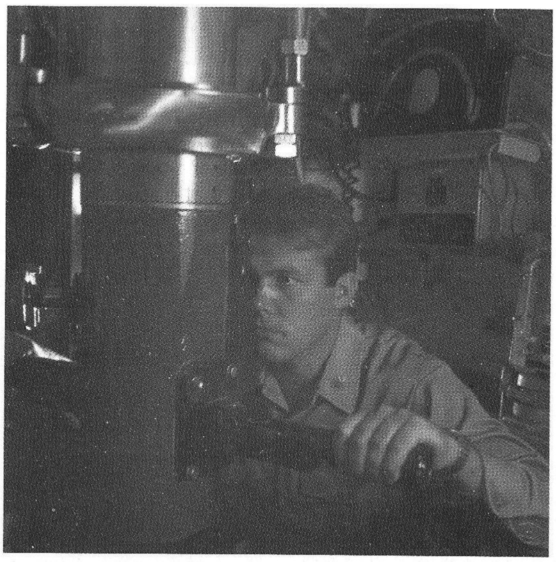
Abacus

Het Amerikaanse softwarebedrijf Abacus komt met een aantal softwarepakketten en boeken voor de Amiga op de markt.
Textpro is een makkelijk te gebruiken tekstverwerkingsprogramma. Het heeft een automatische afbreekroutine. Grafische elementen kunnen in de tekst opgenomen worden.In aanvulling op de pull-down menus heeft de meer ervaren gebruiker ook de beschikking over tal van keyboard commando's.
De verkoopprijs ligt in de VS op $80.
De professionele tegenhanger van Textpro is BeckerText. Naast de standaardopties, die ook te vinden zijn bij de meeste wordprocessors, is hier ook WYSIWYG formattering mogelijk. Grafieken kunnen in de tekst weergegeven worden. BeckerText komt in februari beschikbaar en gaat rond de 150 dollar kosten.
DataRetrieve is een data base manager. DataRetrieve kan datavelden in verschillende stijlen en groottes weergeven, kan records tot 64000 tekens hanteren, verwerkt numerieke waarden tot 15 getallen. Prijs ongeveer $80.
AssemPro is een machinetaalpakket, speciaal ontwikkeld voor de Amiga gebruiker. Amiga programma's kunnen ermee in snelle machinetaal ge-

Duikbootsimulatie voor de Amiga

Klaar voor onder water!

Als je wel eens een oorlogsfilm hebt gezien waarin een duikboot een belangrijke rol speelt dan zul je de geluiden die typisch bij zo'n boot horen niet snel vergeten. De angstige spanning die wordt veroorzaakt door volkomen stilte. Alleen zo af en toe een zacht 'ping', 'ping'... van de sonar van een destroyer die naar je op zoek is. Helemaal beroert word je als het aanzwellende motorgeluid van de duikbootbestrijder pakweg 60 meter boven de duikboot stopt.
Meestal kun je even later de dreunende klappen van de dieptebommen verwachten. En meestal kun je dan ook je maatjes voor de laatste keer vaarwel zeggen.
Je weet niet wat je meemaakt, als je de eerste keer probeert zo diep mogelijk onder water te verdwijnen om aan de wraak van een half verwoest konvooi te ontkomen. Net heb je drie schepen van een Japans konvooi naar de zeebodem geholpen of er maakt zich een duikboot-bestrijdingsvaartuig uit de resten van het scheeps-konvooi los. Met verbazingwekkende snelheid komt dit tot de tanden gewapende vaartuig recht op je af. Bijna dertig knopen snelheid staan er tegenover jouw twintig. En als je duikt ben je zelfs niet sneller dan tien knopen.
Hoe dan wel te ontkomen? Of gewoon het boordkanon gebruiken en de strijd manmoedig aangaan? Dat kun je het beste zelf uitzoeken in deze prima duikboot-simulatie **SILENT SERVICE** van *MicroProse*.
Dit zeer populaire spel is er nu ook voor de Amiga 500. Vele situaties die zich in de tweede wereldoorlog hebben voorgedaan kun je met deze simulator 'naspelen'. Een spannend onderdeel van dit spel is de patrouille over de Pacific oceaan.
Plotseling licht het scherm rood op ten teken dat je 'op de vijand' bent gestoten. Het spel van stil besluipen, wachten, langzaam naderen en beschieten kan beginnen. De nagebootste situaties gaan allemaal over de grote successen die de Amerikanen met hun duikboten op de Japanse vloot behaalden. Wie de eerste keer als captain de motoren van deze submarine laat starten moet voorlopig op hele grote verliezen rekenen.



Verlaat de brug!

Ook een duikboot-simulatie voor de Amiga 500, maar veel en veel moeilijker, is **THE HUNT FOR RED OCTOBER** van *Argus Press Software*. Een zenuwslopende tocht van de allernieuwste Russische duikboot Red October. Het verhaal bij deze simulator is prachtig.
Als kapitein van deze superstille, met de nieuwste (nucleaire) motoren en een ongelooflijk goed verdedigingssysteem uitgeruste duikboot, besluit je naar Amerika te varen. Verraad dus! Het is alleen heel vervelend dat je dat nog niet direct aan de Amerikaanse marine kunt duidelijk maken. Je vaart nog midden tussen de Russische schepen. Het is niet te verwachten dat deze positief zullen reageren als je zomaar koers zet naar de US. Om over de honderd-koppige bemanning die je aan boord hebt nog maar te zwijgen. Aan de andere kant mag je ook niet verwachten dat de Amerikaanse vloot met open armen een zwaar bewapende Russische duikboot hun kust zullen laten naderen. Problemen dus.
Het is dus zaak je boot eerst onopgemerkt door het netwerk van afluisterapparatuur (van de NATO) te sturen. Vervolgens moet je, zonder argwaan bij je navigatie-officier en de bemanning te wekken, richting Amerika varen. Intussen blijf je natuurlijk ver van eventuele Russische pottenkijkers, die de opdracht hebben koste wat kost te voorkomen dat hun paradepaardje in 'verkeerde' handen valt. Als laatste moet je de Amerikanen overtuigen van je goede bedoelingen.
Zoek je weg op prachtig getekende zeekaarten en hou vooral rekening met de daarop in beeld gebrachte (on)diepten. Heb ik het al over mijnen gehad?

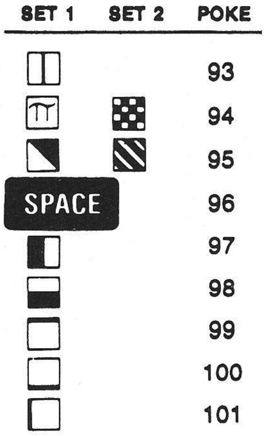
Beide spellen zijn in Nederland uitgebracht door Homesoft Benelux in Haarlem en kosten respectievelijk 89 en 79 gulden.
Rob Timmer

schreven worden. Prijs zo'n $100.
In februari komen van Abacus enkele Amiga boeken uit.
'Amiga BASIC-Inside &Out' is een gids voor het programmeren in Basic. Alle commando's staan beschreven en onderwerpen als graphics, muziek, file management komen uitgebreid aan de orde. In het boek zijn diverse programma's opgenomen, zoals videotitels, grafieken voor statistische weergave van gegevens, vensters, bestandsbeheer en geluidssynthese. Het boek telt 550 pagina's. ISBN 0-916439-88-9.
'Amiga Tricks & Tips' is een collectie kleine programma's. De lezer kan met de beschreven technieken de Amiga mogelijkheden uitgebreid benutten. Aan de orde komen Amiga Basic en C, gebruik van CLI, DOS en de disk drive, programmeren van graphics met behulp van vensters en menu's, etc.
275 pag. ISBN 0-916439-86-7.
'Amiga for Beginners' is een introduktie in het gebruik van Intuition, de muis, de CLI en Amiga Basic. 200 pag. ISBN 1-55755-021-2.
'Amiga Machine Language' is een praktische gids voor iedereen die de Amiga programmeert in snelle machinetaal. Opgenomen is een beschrijving van de 68000 processor, modes en instructieset. Verder nog aandacht voor Amiga libraries voor het gebruik van AmigaDOS, Intuition en geluidsmogelijkheden.

Tekenen en animatie

Anakin Research Inc. uit Toronto heeft een tekentablet voor de Amiga 500 en 2000 uitgebracht. Het bedrijf is gespecialiseerd in soft touch toepassingen, handschrift invoer en handtekening herkenning. Met het drukgevoelige tekentablet 'Easyl' kunnen op papier gezette tekeningen worden overgenomen en in de Amiga worden gevoerd. Anakin Research is ook bezig voor Amiga soft-touch control-programma's te ontwikkelen voor database-, spreadsheet-, CAD- en graphic-programma's.
Byte by Byte uit Austin komt met een zeer krachtig drie-dimensionaal animatie programma. Met 'Sculp-3D' kunnen de animaties worden gemaakt, met schaduw en al, met 'Animate 3D' kan de beweging worden opgezet. Het hele pakket kost ongeveer 250 dollar. Sculp-3D is ook los te gebruiken en kost 100 dollar. Ook heeft het bedrijf Byte by Byte een 'Byte Box' ontworpen. De externe geheugen uitbreiding voor de Amiga 500 kan met deze box oplopen tot 2 MegayBte. Die laatste uitbreiding kost dan wel 700 dollar.

In deze serie lessen wordt aan de lezer stap voor stap een entree geboden in de wereld van de Basic-programma's. We zijn 19 afleveringen terug begonnen met de meest eenvoudige Basic-opdrachten, maar zo simpel is het inmiddels niet meer. Toch kan ook de minder ervaren computerbezitter rustig aan deze cursus meedoen. De programma's zijn allemaal voorzien van een duidelijke uitleg en instructie, waardoor er niet alleen een aardig stuk software wordt gebouwd, maar ook nog het één en ander kan worden geleerd. Deze aflevering gaat over bepaalde Basic-opdrachten in verband met de verwerking van strings.



Basis Basic

Deel 19: Strings (1)

Een belangrijk onderdeel van het hele werken met de Basic interpreter wordt gevormd door 'strings', of wat we voor het gemak 'alfanumerieke gegevens' noemen. Deze data hebben in het gebruik zowel voor- als nadelen, maar een goed programma zal altijd gedeeltelijk bestaan uit stringhandling. We gaan in deze aflevering wat dieper in op bepaalde Basic-opdrachten, die specifiek voor string-afhandeling zijn bedoeld.

Database

De laatste afleveringen zijn we nogal druk in de weer geweest de laatste hand te leggen aan onze eigen database. Dat dit een redelijk succes is geworden blijkt uit de reacties van lezers die ons verblijden met diverse verbeteringen en alle denkbare uitbreidingen op de oorspronkelijke listing.
Hoewel we zeer tevreden zijn over het welslagen van dit project, geven we onmiddellijk toe, dat er natuurlijk talloze varianten zijn te bedenken op de wijze waarop wij dit database probleem hebben aangepakt. De manier waarop de verschillende subroutines zijn geschreven is echter in eerste instantie bedoeld, om ook de wat minder ervaren programmeurs inzicht te geven in de opbouw en afwerking van een programma. Zeker waar het inkortingen betreft zal onze database tot 60 procent van de huidige grootte zijn terug te brengen. Maar daar gaat het niet om. Wel zullen we op de redactie bekijken, of er in de toekomst niet een speciale aflevering in deze cursus kan worden gewijd aan alle door de lezers bedachte varianten.
We nodigen daartoe iedereen uit, zijn gewijzigde of eigen routines in te sturen die betrekking hebben op onze Commodore-Info databasic listing.
Mettertijd kunnen we dan hopelijk kiezen uit een serie van de meest uiteenlopende routines. In ieder geval alvast hartelijk dank voor de medewerking.



Strings

Het fenomeen 'string' zal aan iedere (Basic)programmeur bekend zijn. Zelfs de eerste programma's in alle Basic-boeken beginnen met de verwerking van een string.

```
10   REM TESTPROGRAMMA
20   PRINT "HALLO"
30   END
```

Dit is voor wie dan ook gesneden koek. Toch zit er aan een string, en de afhandeling van deze data nogal wat extra werk voor de computer. Hoewel we dat als programmeur niet altijd in de gaten hebben, bezorgen we onze machine heel wat last, als we een opdracht als deze laten uitvoeren. Voor alle duidelijkheid zullen we eerst wat nader ingaan op wat een string nu in feite is.

Een string is een array van Bytes, waarvan de inhoud bestaat uit een verzameling ASCII-tekens (of karakters) die in de computer worden behandeld en verwerkt als statische data.

Voor de Basic-programmeur is een string niet anders dan een serie karakters die bij elkaar horen. Voor iedere tekst die op scherm of printer wordt gedumpt, gebruiken we strings. De stringvariabele in Basic wordt daarbij onderscheiden van alle andere types variabelen, door het dollarteken ($) achter de naam te plaatsen.
Wat betreft consistentie is gek genoeg de ouderwetse Atari-Basic de beste volger van de originele filosofie. Daar moeten de te gebruiken strings vooraf worden gedeclareerd, compleet met (maximum) lengte en worden ze ook werkelijk behandeld als een array. Dit

vinden we ook terug bij talen als bijvoorbeeld 'C' en C++. 't Is even wennen, maar daana gemakkelijk en efficiënt in het gebruik.
Voor de Commodore gelden andere regels, die op zich als standaard Basic-regels voor strings kunnen worden beschouwd. Variabelen als A$, B1$, A$(3,5), NAAM$(2) enzovoort zijn volledig acceptabel voor de Commodore machines.
Ten opzichte van de numerieke data is het zo, dat het merendeel van de (Basic) operaties die op numerieke variabelen en konstanten kan worden uitgevoerd ook van toepassing is op het werken met strings. Zo maakt het voor opdrachten als READ, INPUT, PRINT, LET niets uit, of we er een floating-point-, integer- of string-naam achter plaatsen. Het is in het laatste geval zelfs zo, dat ook numerieke data door stringvariabelen kunnen worden verwerkt.
Laten we daarvoor het volgende programma bekijken:

```
10   REM LEZEN NUMERIEK (FLP)
100  FOR I = 0 TO 10
120  READ A
130  PRINT A
140  NEXT
200  END
500  REM DATA DATA
510  DATA 123,-444,12, 2225, 65465
520  DATA 200,500,-32770,0,544,777
```

Tot zover zullen we geen problemen ontmoeten tijdens het runnen van deze listing.
Gaan we echter in dit programma het één en ander wijzigen, dan genereren we daarmee een probleem. Kijk maar eens naar de volgende, kleine 'aanpassing':

```
10   REM LEZEN NUMERIEK (INT)
100  FOR I = 0 TO 10
120  READ A%
130  PRINT A%
140  NEXT
200  END
500  REM DATA DATA
510  DATA 123,-444,12, 2225, 65465
520  DATA 200,500,-32770,0,544,777
```

De kenners zullen onmiddelijk het hangijzer ontdekken, waarop dit programma vastloopt. Het is namelijk onmogelijk in een integer variabele (A%) een getal in te lezen dat groter is dan 256*256/2=32767 of kleiner dan 32768. En in de dataregels 510 en 520 treffen we deze getallen aan. Een zelfde verschijnsel valt te genereren bij de eerste listing, door daar de dataregels enigszins te wijzigen:

```
500  REM DATA DATA
510  DATA 123,"-444",12, ABC,
     65465
520  DATA 200,500,"32770",0,
     44,777
```

Bij het runnen zullen we nu vrijwel onmiddellijk een 'DATA TYPE MISMATCH' ontmoeten, omdat we proberen in de variabele A een string in te lezen met de inhoud "-444". Daarbij geven de aanhalingstekens aan, dat het in dit geval een string betreft. Ook de datacomponenten "ABC" en "-32770" zullen een identieke fout oproepen in de Commodore.
De oplossing hiervoor ligt in de volgende listing, waarbij alles in strings wordt ingelezen:

```
10   REM LEZEN STRINGS
100  FOR I = 0 TO 10
120  READ A$
130  PRINT A$
140  NEXT
200  END
500  REM DATA DATA
510  DATA 123,-444,12, 2225, 65465
520  DATA 200,500,-32770,0,544
```

Wat daarbij het meest opvalt, is het feit, dat het niet direct nodig is, om aanhalingstekens te gebruiken in de DATA-regels. Iets anders is, dat de 'getallen', zoals we ze in A$ inlezen, niet kunnen worden verwerkt als echte getallen. We kunnen met deze strings niet veel meer doen, dan alleen printen.
Maar er bestaan uiteraard nogal wat opdrachten om meer te kunnen doen met deze string-variabelen.

String-data

In het voorgaande hebben we kunnen zien, dat het niet direct nodig is, om dataregels, die als stringvariabelen worden verwerkt, te voorzien van aanhalingstekens. Data uit een hypothetische database als:

```
500  DATA HARRY,12,3,Telex,12345
510  DATA SANDRA,6,89,
     Telefoon,54321
520  DATA MARLOES,11,64,
     Telefax,77346
```

zullen keurig worden verwerkt, ondanks het ontbreken van de aanhalingstekens. Anders ligt dit, als bepaalde gegevens in de string bijvoorbeeld moeten worden geformatteerd. We willen bijvoorbeeld een fraaie uitlijsting maken, waarbij alle strings evenlang moeten zijn. Hoewel er tijdens het printen mogelijkheden bestaan dit in één regel te kunnen doen, willen we de rudimentaire manier niet overslaan:

```
500  DATA "HARRY    "," 12","
     3","Telex    "," 12345"
510  DATA "SANDRA   "," 6","
     89","Telefoon "," 54321"
520  DATA "MARLOES  "," 11","
     64","Telefax  "," 77342"
```

In proportionele druk, zoals in dit blad, is het moeilijk om de juiste spatiëring te tonen, maar de bedoeling zal duidelijk zijn. Alle verschillende DATA-stukjes hebben per groep dezelfde lengte. Om dit te bewerken moet de data tussen aanhalingstekens worden gezet. Ook als er sprake is van bijvoorbeeld komma's in de data zelf is het nodig de string tussen aanhalingstekens te zetten.

LEN

We hebben het al even over de lengte gehad, van een string. De tekst "ABCDEFG" is welgeteld 7 karakters lang. We kunnen dit eenvoudig vaststellen, door de afzonderlijke karakters te tellen. Het toekennen van deze string aan een variabele is niet moeilijk:

```
10   A$="ABCDEFG"
```

Bij de computer ligt het bepalen van de lengte van een string wat anders. We moeten een functie gebruiken om dit te kunnen doen. Deze functie heeft de naam LEN. Zoals bekend bestaat er verschil tussen een functie en een opdracht. LEN als functie heeft altijd een Basicopdracht of variabele nodig om in Basic te kunnen worden geïmplementeerd:

```
20   LEN(A$)
```

zal op een Syntax error vastlopen, in tegensterlling tot:

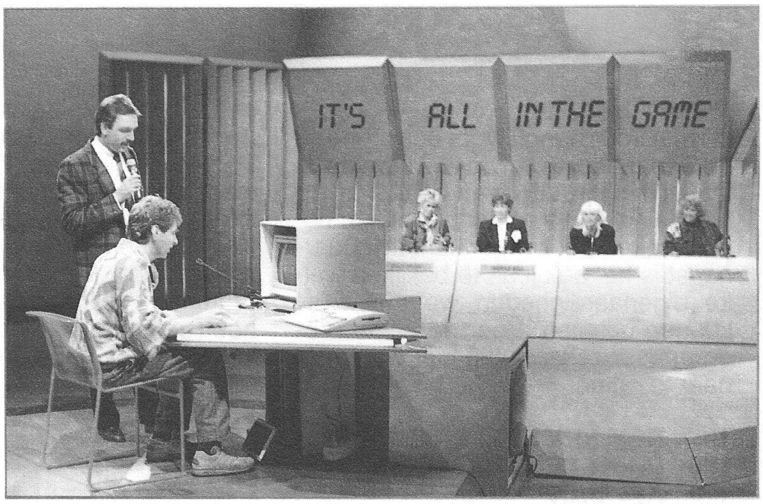
```
20   A = LEN(A$) of

20   PRINT LEN(A$)
```

In ons geval krijgen we bij LEN voor "ABCDEFG" in variabele A de waarde 7 (let er op dat de aanhalinggestekens niet worden meegeteld).

Computerspelletjes winnen nog steeds aan populariteit. Dat blijkt ook uit het spelprogramma 'It's all in the Game' van de NCRV. Een uniek samenwerkingsprojekt gaat er nu voor zorgen, dat de TV-games uit dat programma ook thuis kunnen worden gespeeld.

It's all in the Game



Presentator Henk Mouwe met enkele kandidaten.

De NCRV, René Stokvis Produkties en Sala Communications hebben gemeenschappelijk een uniek projekt opgezet. De computergames, die dit spelprogramma de revue zullen passeren, kunnen op diskette besteld worden, waardoor de kijkers nu ook zelf aktief met de spelletjes 'in de slag' kunnen.

It's All in the Game is vanaf dit seizoen elke maand op maandagavond te zien na het journaal van acht uur. Presentator Henk Mouwe test dan de behendigheid van een achttal kandidaten (in twee teams) op het gebied van computerspelletjes. De formule blijkt uitstekend aan te slaan getuige de kijkcijfers.

Thuis spelen

Om meer mensen de mogeljkheid te bieden de spelletjes te spelen, is nu besloten de games ook op een diskette uit te brengen. In overleg met René Stokvis Produkties (de bedenkers van dit programma) en de NCRV zal Sala Communications (uitgever van diverse computer-magazines, waaronder Commodore Info) de produktie en verkoop van diskettes gaan verzorgen. Zo kunnen ook de thuisblijvers en de fervente computergame-spelers hun eigen kwaliteiten testen met dezelfde spelletjes als in de programma's aan de orde zullen komen.

Diskette 1 (Alleen voor C-64/128)

Op de eerste diskette staan de volgende spelletjes:

° **Balletje de Luxe**
één balletje, zes dopjes, en maar raden...

° **Schuif Schuif**
3 logo's, 8 dopjes. Schuiven in een oogwenk.

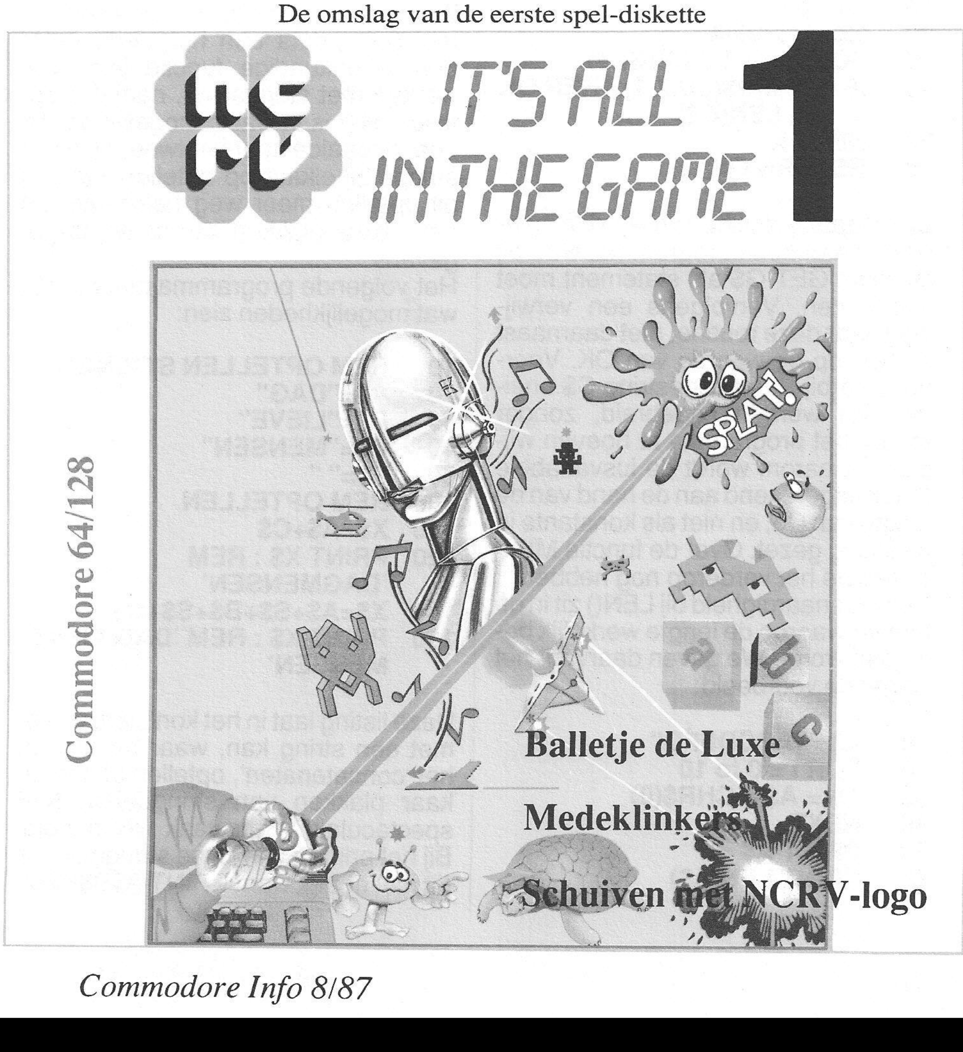
° **Medeklinkers**
16 cotegorieën, vele honderden woorden, maar... géén medeklinkers!

Aanvragen

Bestellen van de diskettes met daarop de games kan eenvoudig door ƒ19,95 over te maken op gironummer 158549 t.v.n. Sala Communications te Amsterdam met vermelding: 'It's all in the game'. U kunt ook een girobetaalkaart opsturen naar Postbus 5570, 1007 AN Amsterdam (ook hierop niet vergeten te zeggen, welke diskette u wilt ontvangen).

Natuurlijk kunt u voor al deze informatie ook naar het programma kijken, dan heeft u dubbel plezier!

De omslag van de eerste spel-diskette



De maximum lengte voor een string bij de Commodores ligt op 256 karakters. Op zichzelf is LEN een hele simpele functie, waar we echter in de Commodore zeker niet zonder kunnen. Daarnaast is het ook leuk te bekijken welke eigenaardigheden we met LEN kunnen uithalen. Voor de echte programmeur zeker nuttig hier wat beter naar te kijken:

```
10  REM LEN SPELERIJ
100 PRINT LEN("HALLO")
110 A$="JAN"
120 B$="PIET"
130 PRINT LEN(A$)
140 PRINT LEN(A$+B$)
150 PRINT LEN(A$)+3
160 PRINT SPC(20 - LEN(A$));A$ :
    REM RECHTS UITLIJNEN
180 PRINT TAB(80 - LEN(A$+"
    "+B$)/2);A$;" ";B$: REM
    CENTREREN PRINTER
190 :
200 IF (LEN(A$)) 10 THEN PRINT
    "Maximaal 10 karakters"
```

Als laatste een kontroleprogramma op bepaalde kodes, waarvan er één door de gebruiker moet worden ingetypt:

```
10  REM KODE-TEST
15  OK=0
20  C$="#$%@&"
30  FOR X = 1 TO LEN(C$)
40  IF G$=MID$(C$,X,1) THEN OK=
    1 : X = LEN(C$)
50  NEXT X
60  RETURN OK
```

Deze laatste routine kan worden ingepast in een hoofdprogramma, waarbij de regel GET G$ als statement moet voorkomen. Vervolgens een verwijzing naar deze routine, met daarnaast de test op de waarde van OK. Voordeel hierbij is dat codestring C$ oneindig kan worden uitgebreid, zonder verder het programma te hoeven wijzigen. Daarom wordt de lusvariabele X ook uitgerekend aan de hand van de lengte van C$, en niet als konstante in de listing gezet. Over de functie MID$ zullen we het verderop nog hebben.
Een eigenaardigheid bij LEN() zit in de manier waarop de lengte werkelijk berekend wordt. We geven daarvoor het volgende voorbeeld:

```
10 A$= "ABCDEFG"
20 FOR I = 0 to 10
30 A$ = A$ + CHR$(0)
40 NEXT
50 PRINT A$
60 PRINT LEN(A$)
```

Als A$ zien we uitsluitend "ABCDE FG" op het scherm, 7 karakters dus, maar de LEN() functie vertelt ons de totale lengte, compleet met 11 CHR$ (0) karakters, die we niet op het scherm zien. Een fout, die menig programmeur de nodige hoofdbrekens heeft gekost, zeker, omdat alle controle-karakters in LEN() worden meegeteld, dus ook RVSON etcetera. Zeker een bug om rekening mee te houden.

SET 1	SET 2	POKE	SET 1	SET 2	POKE
		105			117
		106			118
		107			119
		108			120
		109			121
		110		✓	122
		111			123
		112			124
		113			125
		114			126
		115			127
		116			

Optellen

We kunnen niet direct mathematisch met string-data aan het werk, maar één rekenkundige functie bezit het werken met strings wel, namelijk optellen. Het is in Basic mogelijk om op een bepaalde manier twee of meer strings bij elkaar op te tellen, waarbij dit optellen meer weg heeft van het aan elkaar plakken van de string-gegevens.
Het volgende programma laat daarbij wat mogelijkheden zien:

```
10  REM OPTELLEN STRINGS
20  A$="DAG"
30  B$="LIEVE"
40  C$="MENSEN"
50  S$=" "
100 REM OPTELLEN
110 X$=A$+C$
120 PRINT X$ : REM
    'DAGMENSEN'
140 X$=A$+S$+B$+S$+C$
150 PRINT X$ : REM 'DAG LIEVE
    MENSEN'
```

Deze listing laat in het kort zien, wat er met een string kan, waar het betreft het 'concatenaten', optellen of aan elkaar plakken van stringdelen. Niet spectaculair, maar vaak wel handig. Bij het printen van losse stringdelen is dit niet nodig, want PRINT ASBS C$ doet precies hetzelfde als regel 140 en 150, maar daarbij hebben we nog geen waarde in X$. Dat moet gebeuren op bovenstaande manier en resulteert daarbij in een gezegde van één van 's lands bekende zangers uit vroeger dagen.
Het aftrekken, delen, vermenigvuldigen en machtsverheffen met strings is helaas niet mogelijk.
Bij het optellen van stringdelen moeten we er voor zorgen, dat de totale string een lengte krijgt kleiner dan de maximaal toelaatbare stringlengte van 255 karakters.

ASCII waarde

We kunnen gelukkig naast het optellen ook aan de slag met het rangschikken van strings. Daarbij speelt de ASCII waarde van de in de string aanwezige karakters een grote rol. Op grond van deze waarde wordt door de computer de 'waarde' van de string bepaald. Hoe groter de 'waarde' hoe verder achtering de string zal worden gesorteerd.

In ons Basic database-programma hebben we bij het sorteren met Quicksort gebruik gemaakt van deze ASCII stringwaarden. Voor de duidelijkheid ook hier een paar voorbeelden in een Basic-programma:

```
10  REM STRING vergelijking
20  PRINT "A$ ";
30  INPUT A$
40  IF A$ = "STOP" THEN END
50  PRINT "B$ ";
60  INPUT B$
100 IF A$ > B$ THEN 200
110 IF A$ < B$ THEN 250
150 PRINT "A$ gelijk aan B$"
160 GOTO 20
200 PRINT "A$ groter B$"
210 GOTO 20
250 PRINT "A$ kleiner B$"
260 GOTO 20
```

Door bij A$ en B$ diverse strings in te typen, wordt duidelijk, dat aan de hand van de ASCII-waarden, die zo gekozen zijn, dat de waarden corresponderen met ons alfabet, de strings kunnen worden gerangschikt. Ook verderop in de string kan de vergelijking worden uitgevoerd. Geef voor A$ en B$ bijvoorbeeld respectievelijk de waarden: "AAAAAAABZ" en "AAAAAAACX" en zie wat er gebeurt. De afzonderlijke ASCII (American Standard Code for Information Interchange) codes -waar Commodore zich niet

helemaal aan houdt- kunnen we per karakter ook op een andere manier te weten komen. Daarvoor hebben we de functie ASC().

ASC()

Deze functie geeft als numeriek een 'Commodore' ASCII waarde retour die overeenkomt met het eerste karakter uit de bepaalde string.
ASC() is nodig bij het testen van individuele karakters. Vooral bij schermmanipulaties en toetsenbord-input is het nodig om van deze ASC() functie gebruik te maken.
Ook is het in bepaalde gevallen sneller en simpeler te programmeren met ASCII-waarden dan de string-equivalent. Dit laatste gaat bijvoorbeeld op bij een 'JA/NEE' routine, waarbij input van het toetsenbord wordt gevraagd als "J" of "N". Bij het testen van de input kun je dan in principe kiezen tussen string en numerieke vergelijking. Aan de hand van een listing is het verschil duidelijk te zien:

```
1000 REM INPUT-ROUTINE
1010 REM J=74 / N= 78
1020 PRINT " J(A) / N(EE) ";
1030 GET A$: IF A$="" THEN 1030
1040 KY=ASC(A$)
1050 IF KY<>74 AND KY <>78
     THEN 1020
1060 RETURN
```

Op deze manier 'vertalen' we als het ware de waarde van de input in een numerieke variabele (KY), waarmee gemakkelijker de test is uit te voeren. Het string alternatief ziet er zo uit:

```
1000 REM INPUT-ROUTINE
1010 REM J=74 / N= 78
1020 PRINT " J(A) / N(EE) ";
1030 GET A$: IF A$="" THEN 1030
1040
1050 IF LEFT$(A$,1)<>"J" AND
     LEFT$(A$,1)<>"J" THEN 1020
1060 RETURN
```

Deze stringtest is gemakkelijker te lezen (en dus te debuggen) dan de 'ASCII' listing van dezelfde routine, maar daarnaast moeten we met meer en andere stringfuncties aan het werk (LEFT$) om hetzelfde doel te bereiken. Korter en dus beter is het om de ASC() functie toe te passen.
Er zitten een paar addertjes onder het gras, waar het de implementatie van deze functie op de meeste Commodore computers betreft. Zo kun je ASC() niet loslaten op een string met niks. Dus A$="": PRINT ASC(A$) stuit op de melding '?ILLEGAL QUANTITY ERROR'. Vandaar ook het blok in regel 1030 van de vorige routine, waarbij direct de waarde 'niks' ("") wordt afgevangen.
Andere Basic-dialecten doen dit beter, door aan een string met "" de waarde 0 toe te kennen. Bij Commodore moeten we dit allemaal zelf doen. We kunnen dit uiteraard gemakkelijk bewerken, door in plaats van bovenstaande regel 1030 te schrijven:

```
1030 GET A$ : IF A$ = "" THEN A$
     = CHR$(0)
```

Dit heeft exact hetzelfde effect en leidt de interpreter van de Commodore om de 'bug' van de ASC() functie heen.

CHR$()

De tegengestelde functie van ASC() is een functie met de naam CHR$(). Deze functie converteert iedere numerieke waarde tussen 0 en 255 naar een string met lengte 1 en als inhoud het CBM-ASCII equivalente karakter.
Met deze functie is het mogelijk om bepaalde karakters, zoals bijvoorbeeld TAB, BACKSPACE, RETURN, aanhalingsteken en ESCAPE op te nemen in een string, ermee te manipuleren en te printen.
Aan de hand van de ASCII-tabel is het niet moeilijk te bepalen welke waarden aan CHR$() moeten worden gegeven om bepaalde karakters te genereren. Zo geeft CHR$(13) een RETURN terwijl CHR$(34) kan zorgen voor aanhalingstekens. Voor niet gehele getallen vindt bij CHR$() een afronding plaats, terwijl getallen buiten de range van 0 tot 255 voor een foutmelding zorgen.
CHR$(500), CHR$(-2) en CHR$(A$) zullen dus niet door de Commodore worden uitgevoerd, terwijl CHR$(.2), CHR$(ASC("#")) en CHR$(255.9) tot een geldige evaluatie leiden.
Ook CHR$(7) is een leuk karakter om mee te werken. Daardoor zal zowel op de printer als in de computer tijdens het printen de bel gaan rinkelen. CHR$() kan ook goed worden toegepast bij het vertalen van 'gePEEKte' codes, waarbij de PEEK een karakter voorstelt. Ook het maken van relatief eenvoudige beveiligingen is mogelijk met hulp van CHR$(), omdat bijvoorbeeld CHR$(0) wel een karakter toevoegt aan de lengte van een string, maar niet zichtbaar is op scherm of printer. Zie daarvoor onder ASC(). Als afsluiting een paar toepassingsregels met CHR$():

```
10   X$ = CHR$(34) + CHR$(7) +
     CHR$(18) + "NAAM" +
     CHR$(146) + CHR$(34)
```

De string NAAM wordt hier voorzien van aanhalingstekens en de karakters om deze string met BELL en RVS/RVSOFF te printen.

```
20   CL$ = CHR$(147) : REM Clear
     Screen
30   PRINT CHR$(7)   : REM ring
     ring
40   ES$=CHR$(27)    : REM
     ESCAPE/
```

De escape kan worden gebruikt voor het aansturen van niet Commodore-printers. De meeste controlecodes voor normale printers duiden stuurcoderingen aan door het ESCAPE-karakter als voorvoegsel mee te geven.

Tot slot

Tot zover deze wat uitgebreide aflevering specifiek over strings. De volgende keer zullen we doorgaan met dit onderwerp, waarbij we onder meer aandacht zullen besteden aan functies die rechtstreeks manipuleren met de inhoud van een string. Sterkte en veel programmeerplezier. Reacties en opmerkingen zijn altijd welkom op het redactie-adres.

Jan Bodzinga

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
D	68	♠	97		126	Grey 3	155
E	69		98		127	PUR	156
F	70		99		128	CRSR	157
G	71		100	Orange	129	YEL	158
H	72		101		130	CYN	159
I	73		102		131	SPACE	160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112	SHIFT RETURN	141		170
T	84	●	113	SWITCH TO UPPER CASE	142		171
U	85		114		143		172
V	86	♥	115	BLK	144		173
W	87		116	CRSR	145		174
X	88		117	RVS OFF	146		175
Y	89	⊠	118	CLR HOME	147		176
Z	90	○	119	INST DEL	148		177
[	91	♣	120	Brown	149		178
£	92		121	Lt. Red	150		179
]	93	♦	122	Grey 1	151		180
↑	94	┼	123	Grey 2	152		181
←	95		124	Lt. Green	153		182
	96		125	Lt. Blue	154		183

We schieten aardig op met onze cursus Machinetaal. De vorige aflevering was in z'n geheel gewijd aan onze nieuwe assembler, en deze keer zullen we de machinetaal-software eens aan de tand voelen.

Cursus Machinetaal

door Tjipke van der Land



Deel 10: Assembler

Het programmeren in machinetaal is in theorie mogelijk zonder verdere hulpmiddelen. Maar veel handiger is het, om gebruik te maken van een assembler/disassembler waarmee we de binaire machinetaal-codes in wat beter leesbare taal kunnen verwerken. De computer zelf komt ons daar gelukkig bij te hulp.

Deze keer gaan we door met de assembler. De voordelen hiervan zullen inmiddels wel duidelijk zijn en daarom gaan we gelijk maar eens kijken hoe de assembler werkt. Hopelijk is het intypen van de volledig listing gelukt. Zo niet, dan bestaat de mogelijkheid om via Infolist een tape of disk met de listing aan te schaffen. Voor het werken van de assembler moet je een aantal commando's kennen die alleen maar betrekking hebben op de assembler. Deze opdrachten worden dus uitsluitend gebruikt om de Assembler te besturen en hebben geen enkel effect op het uiteindelijke programma.
Bijvoorbeeld hoe kan ik een programma en een list opvragen van de op dit moment aanwezige machinetaalcode? Hoe kan ik mijn urenlange arbeid op papier krijgen?
Dit zijn allemaal vragen, waar je zo 1, 2, 3 niet op komt, maar pas wanneer je vurig aan het programmeren bent worden ze evident. Dus is het verstandiger om eerst even te bekijken hoe deze kommando's werken, voordat je een heleboel werk voor niets doet. En dit is wat we juist willen voorkomen met onze mini-assembler.

Opdrachten

In de lijst hiernaast vindt je alle kommando's die we nodig hebben voor het programmeren. We gaan niet alle kommando's op het rijtje af behandelen, omdat dat eigenlijk een beetje saai is. We gaan gewoon kijken naar wat de meestgebruikte instructies zijn, om een programma te kunnen maken.

Om te beginnen moeten we dus machinetaal instructies in de computer stoppen (zie afb. 1).
Stel we willen een programma invoeren dat begint op een adres $ 1000. De instructies die we willen gaan invoeren, en die door de mini-assembler worden omgezet in voor de computer begrijpelijke getallen, moeten worden geassembleerd, dus is het logisch dat onze eerste assembler-instructie de A is van assembleren. Op het scherm zien we een punt (.) als de assembler is ingeladen en via SYS 49152 is gestart, als teken, dat we een opdracht in kunnen typen.
Het begint dus allemaal met de "A" met daarachter het adres waar je wilt beginnen. Daarachter komt de machinetaali-nstructie die we willen gaan gebruiken. Bijvoorbeeld:

.A $1000 LDA #$17

Je ziet als je op de RETURN toets drukt dat je daarna de volgende instructie kunt invoeren, omdat de assembler weet dat LDA #$17 geen derde Byte voor z'n adres nodig heeft. De Assembler kent dit type instructie als een 'immediate'. Je moet toch wel iets meegeven, waardoor de assembler kan weten, dat het geen adres is, maar een instructie waarbij een waarde onmiddellijk (immediate) naar een register moet worden geschreven. Dit doe je door het "#" tekentje voor het hexadecimale getal ($17) neer te zetten. Als je goed kijkt dan zie je ook dat de assembler precies het juiste aantal Bytes reserveert voor de instructie(s) die je hebt ingevoerd. Op deze manier

Afb.1

Instructielijst assembler	Kommando's	
Starten mini-assembler	Bijv	sys 49152 (CR)
A = assembleren	Bijv.	.a 1000 LDA $3000
D = disassembleren	Bijv.	.d 1000
F = Vullen van geheugen	Bijv.	.f 1000 2000 FF
G = Starten van programma	Bijv.	.g 1000
H = Zoeken naar data	Bijv.	.h 2000 2100 'JAN'
L = Laden van Programma's	Bijv.	.l "Jan Kl",08
M = Overzicht geheugen	Bijv.	.m 1000 1050
P = Printen bron programma	Bijv.	.p 1000 1100
R = Afdrukken registers	Bijv.	.r
S = Opslaan van programma's	Bijv.	.s 1000 1060
T = Verplaatsen geheugenblok	Bijv.	.t 1000 2000 4000
X = Stoppen miniassembler	Bijv.	. x

kun je dus gewoon verder gaan met instructies invoeren.

Koek en ei

Er is helaas één 'maar' aan deze prachtige manier van programmeren. Net als bij alle andere wat complexe handelingen geldt ook hier: Bezint eer ge begint.
Je moet beslist geen structurele fout maken, door bijvoorbeeld een immediate-instructie in te voeren terwijl het een absolute instructie moet zijn. Kom je daar een beetje laat achter dan zit je met een probleem, omdat je dan een Byte te kort komt. Dit is heel lastig, want dan moet je rest van je programma een stuk opschuiven, met de assemblerinstructie 'T'. Voor relatieve programma sprongen is dit helemaal niet erg, maar als je echt een jump maakt naar een specifiek adres dan moet je al deze instructies nalopen om ze te verbeteren. Het is dan ook het beste om zoveel mogelijk relatief te gaan springen in een programma.

Relatief gespring

Tot nu toe hebben we het nog niet gehad over relatieve sprongen maken binnen een programma. Maar dat kunnen we in deze les mooi meepakken, omdat dit toch één van de meest gebruikte machinetaal instructies is. Wat we wel hebben behandeld is het echte 'springen' naar een plek door middel van JMP instructie. Deze instructie gaat gelijk naar de plek (het adres) dat je aanwijst. Maar je kunt ook een instructie geven, waar je zegt van spring maar 60 sprongen terug naar een adres of sub-programma wat daar staat. Deze manier van springen doen we meestal aan de hand van bepaalde condities, bijvoorbeeld staat er een flag op nul, of een register is leeg. Op het moment dat je dan een heel stuk programma moet 'moven' (verplaatsen), wat dus kan met de mini-assembler, dan is er in principe geen verschil gekomen in het relatief springen, want dat sub-programma staat nog steeds 60 sprongen terug ten opzichte van het uitgangspunt. De absolute adressen zijn echter wel veranderd, en dan kom je met een absolute sprong-instructie zoals **JMP $ 4501** niet meer op de goede plek. Dit is ook één van de nadelen van een mini-assembler; maar ja, je kunt niet alles hebben.
Inmiddels zijn we weer een beetje afgedwaald van waar we mee bezig waren, namelijk de assembler instructies. Maar desalniettemin moeten we dit goed onthouden, want we gaan het bovenstaande straks uitwerken in een voorbeeld, waardoor we er gelijk weer wat instructie's bij leren.

Bewaren

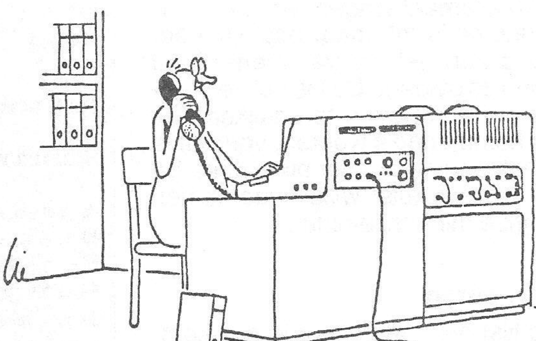
Het belangrijkste wat we nu gaan leren, is het eerst opslaan (SAVEN) van het programma, hoeveel fouten er na de eerste invoer ook in zitten. Waarom doen we dat ?
Dit gebeurt om ons een hoop onnodig tikwerk te besparen. Ondanks dat we inmiddels zielsgelukkig zijn met zo'n geavanceerde mini-assembler, waardoor het een stuk gemakkelijker is geworden, is het overtikken van een hoop denkwerk het beroerdste wat er is.
Want wat gebeurt er nu: na het invoeren van de laatste brandende instructie's die nog nodig waren om het programma te voltooien typen we daarna vlug de verlossende instructie:

.g $1000

wat betekent: Ga naar de start van het programma wat begint op adres $1000 en RUN. En, wat een ontreddering, het programma loopt vast, want we hebben vergeten tijdens het 'moven' een JMP instructie aan te passen.
Heel spijtig allemaal, want nu had je net iets moois bedacht, en dat moet je nu allemaal weer opnieuw bedenken. Heel spijtig !
Het beste wat je in zo'n situatie kunt doen is de computer uitzetten, en maar een stripboekje gaan lezen, want computers zijn toch maar rotdingen. Niets is minder waar, want het is je eigen schuld. Als je de moeite had genomen om het programmeer proces nog even met 1 minuut te verlengen door het programma te saven voor het testen, dan is een computer in één keer geen rotding meer. Dus altijd voor het testen eerst saven.

Het programma:

MNEMONIC	HEX	COMMENTAAR
LDA #$17	A9 17	;Getal laden voor ASCII nivo laag
STA $D018	8D 18 D0	;Parameter plaatsen
LDA $2000	AD 00 20	;Ophalen letter 1
STA $0578	D 78 05	;Plaatsen van letter 1 in videogeh.
LDA $2001	D 01 20	;Ophalen letter 2
STA $0579	8D 79 05	;Plaatsen van letter 2 in videogeh.
LDA $2002	AD 02 20	;Enz.
STA $057A	8D 7A 05	
LDA $2003	AD 03 20	;Ophalen van spatie
STA $057B	8D 7B 05	;Plaatsen van spatie in videogeh.
LDA $2004	AD 04 20	;Enz.
STA $057	8D 7C 05	
LDA $200	AD 05 20	
STA $057D	8D 7D 05	
LDA $2006	AD 06 20	
STA $057E	8D 7E 05	
LDA $2007	AD 07 20	
STA $057F	8D 7F 05	
LDA $2008	AD 08 20	
STA $0580	8D 80 05	
LDA $2009	AD 09 20	
STA $0581	8D 81 05	
LDA $200A	AD 0A 20	
STA $0582	8D 82 05	
LDA $200B	AD 0B 20	
STA $0583	8D 83 05	
NOP	EA	
BNE $104D	D0 FD	
	;LOOP VIA NOP (NO OPERATION)	



Voorbeeld

Hoe het saven, moven en starten van een programma in zijn werk gaat, gaan we nu door middel van een voorbeeld duidelijk maken. Als je de vorige keer al wat met de assembler hebt geoefend, met het voorbeeld van de vorige keer, dan zal het wel een beetje dagen, denk ik. Maar voor de eenvoud beginnen we maar even bij het begin. Na het laden en starten van de assembler staat hij uiteraard startklaar om gebruikt te worden.

Om te assembleren moeten we dus invoeren .a $"begin adres" "instructie", dus:

.A $1000 LDA #$17
of .A $1000 LDA $1000.

Na de return geeft de mini assembler automatisch het veronderstelde volgende adres, waar de eerstvolgende instructie moet komen.
Vervolgens kun je gewoon alle instructies die je wilt invoeren gaan intikken zonder dat je het adres nogmaals hoeft op te geven. Het voordeel is daarbij, dat je niet hoeft te tellen uit hoeveel "statements" (Bytes) iedere instructie bestaat, dat doet gelukkig de mini-assembler.
Voor het stoppen met assembleren voer je geen enkele instructie in, maar geef je gewoon een return op een lege regel. Wil je na het invoeren van het programma, wat zo groot kan zijn dat je het niet meer compleet kunt bekijken op het scherm, toch een volledige listing, dan kun je door het invoeren van. M "begin adres" "eind adres", bijvoorbeeld:

.M 1000 1050

het gedeelte overzien, wat je op dat moment wilt bekijken. Je ziet dan niet de instructies zoals je ze ingevoerd hebt, maar alles zoals het door de mini-assembler is vertaald in hexadecimalen.
Wil je hierin toch nog een verandering aanbrengen, dan kun je gewoon met de cursor naar het gewenste hexadecimale getal en deze wijzigen. Dit kan voorkomen als je bijvoorbeeld een offset-sprong wilt verbeteren of een jump.

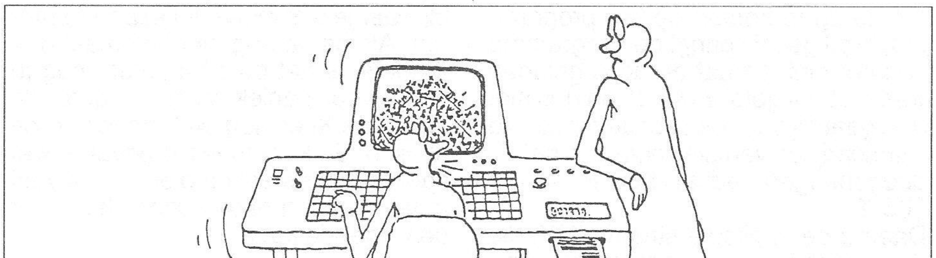
Disassembly

Wil je de ingevoerde instructies weer terugzien zoals je ze ingevoerd hebt, dan kan dat door in te tikken:

.D $1000.

Met de "D" geef je aan dat je wil disassembleren, dus ontleden. Verder werkt deze instructie net als bij de A van assembleren. Als we een programma hebben ingevoerd gaan we eerst het gehhel opslaan. Zorg er voor dat je een geformatteerde schijf in je drive hebt zitten zonder 'write protect', daarna intikken:

(S "jan kl",08,1000,1051)
Je ziet nu dat het programma veilig is weggeschreven. Voor de gebruikers onder ons die een cassette-deck hebben, moeten hetzelfde invoeren, alleen voor de "08" een "01" neerzetten. Voordat we het kunnen testen moeten we nog even de hexadecimale waarden die het woord JAN KLAASSEN vormen in het geheugen zetten. In het voorbeeld van de vorige keer konden we zien dat we POKEten vanaf adres 8192 decimaal, wat dus $2000 hexadecimaal is. Dit is nu veel makkelijker dan de vorige keer want we kunnen nu de waarden rechtstreeks invoeren, door in te tikken:



:2000 4A 41 4E 20 4B 4C 41 41 (return)

vervolgens:

:2008 53 53 45 4E (return)

Je ziet dat we maar 8 Bytes invoeren per regel, meer kan namelijk niet. Je kunt nog even controleren of alles goed is door in te tikken:

.M 2000 2050

waar je de ingevoerde Bytes kunt zien vanaf adres $2000. Daarna kunnen we het programma testen door in te tikken:

.g 1000

Om de letters zichtbaar te houden hebben we het programma even in een lus gezet, want anders zie je de letters maar zo kort (ongeveer 1/50 sec), dat je niet kunt lezen wat je gemaakt hebt. Als het goed is zie je tussen de getallen "JAN KLAASSEN " staan of je eigen naam als je de moeite hebt genomen om die in ASCII over te zetten. Door te drukken op "RUN /STOP RESTORE" kom je terug in de monitor.

Nieuwe instructies

Omdat we nu zover zijn dat we simpele programma's op de monitor kunnen maken, gaan we het zojuist gemaakte programma verbeteren met nieuwe instructies. Je kunt dan ook zien dat met een beetje meer denkwerk veel minder tikwerk kan worden bewerkstelligd. Om dit probleem op te lossen gaan we gebruikmaken van de registers die standaard aanwezig zijn in de 65xx. De registers waar we het over hebben staan in onderstaand schema, wat overigens ook al in een voorgaande uitgave heeft gestaan. De registers die we bedoelen zijn het X en Y register. deze registers gaan we dadelijk gebruiken als teller, om te adresseren. Tot nu toe hebben we gewoon een waarde geladen en opgeslagen. Dit kan veel handiger. Zoals we inmiddels weten zijn er diverse instructie methoden voor één instructie. De methode die we er nu leren is namelijk het geïndexeerd adresseren. Deze manier geldt zowel voor het laden van de accumulator, alsook voor

het opslaan in de accu. Hiervoor is het nodig om nog een paar instructie bij te leren, om ons bovenstaande programma te optimaliseren. We beginnen met het geïndexeerd op register X laden van de accu. Dit betekent in principe hetzelfde als absolute, maar hier wordt de waarde die geladen is in het X register opgeteld bij het absolute adres.
Dus als we de instructie LDA $1000,X laten uitvoeren door de assembler, en de waarde in het X register is op dat moment 5, dan laadt de accu zijn waarde van adres $1000 + $5 in totaal $1005 binnen.
Als we deze instructie op deze manier invoeren, weet de assembler dat hij geïndexeerd op X moet adresseren.
Vervolgens is er de 'implied instructie' INX, die de waarde in het X register met 1 verhoogt.
In het onderstaande programma, wat de geoptimaliseerde versie is van de vorige listing, kunnen we goed het verschil in lengte onderscheiden.

Het programma:

MNEMONIC	COMMENTAAR
LDX #$00	; X REGISTER LADEN MET 0
LDA #$17	;ACCU LADEN MET $17 VOOR ASCII NIVO LAAG
STA $D018	;PLAATSEN PARAMETER
LDA $2000,X	;LAAD ACCU GEINDEXEERD X
BEQ $1013	;SPRING NAAR ADRES $1015 ALS Z FLAG = 1
STA $0578,X	;SLA HET GETAL GEÏNDEXEERD X OP
INX	;VERHOOG X MET 1
JMP $1007	;SPRING NAAR ADRES $1007
BRK	;TERUG NAAR MONITOR

We zullen het programma stapje voor stapje doornemen om het een beetje duidelijk te maken. Op deze manier kunnen we ook de nieuwe instructies, die op zich niet moeilijk zijn even doornemen.

Programma-flow

We beginnen in het programma met het laden van getal 0 in het X register. Dit doen we om er zeker van te zijn dat er ook echt 0 in staat, dit om te voorkomen dat er niet nog een getal in blijft staan van een vorig programma. Vervolgens gaan we de computer instellen met een paar waarden die nodig zijn voor het instellen van kleine letters, zodat Jan Klaassen zichtbaar wordt in de vorm zoals we hem graag zien, en niet in grafische tekens. Dan begint het echte programma, door het getal te laden geïndexeerd X van adres $2000. Omdat de waarde in X 0 is, (zojuist geladen) haalt hij het adres van $2000, omdat $2000 + 0 = $2000. Hier staat het eerste ASCII getal, namelijk 4A.
Dan staat er een nieuwe instructie, namelijk BEQ. Deze betekent Branch if equal to zero, vertaald in nederlands, maak een sprong onder de conditie dat de waarde 0 is. Deze instructie kijkt naar de status flags. Wanneer de Z flag 1 is, wat betekent dat de waarde nul is, gaat hij een relatieve sprong maken. Het relatief springen betekent zoveel sprongen vooruit, of als het getal achter de sprongininstructie negatief is, bijvoorbeeld "FC", dan springt hij even zoveel bytes terug. In de assembler kunnen we het adres opgeven waar we hem naar toe willen laten springen. Het programma (assembler) berekent dan zelf wat de offsetwaarde moet worden. Als je kijkt bij de hex waarde die bij de instructie staat, dan zie je dat dit $07 (positief dus 7 sprongen vooruit). Als je gaat tellen zul je tot de ontdekking komen dat je 1 tekort komt om de BRK instructie te halen. Dit komt omdat de programmateller na het uitvoeren van een instructie zichzelf alvast met 1 verhoogt, omdat hij daar toch de volgende instructie moet halen. Met dit gegeven in ons achterhoofd komen we dus wel op de BRK instructie terecht.
Om terug te komen op het programma, in dit geval springt de Programma counter niet, omdat de accu geladen was met het getal $4A ('J') en dus niet 0 is, gaat hij door met het geïndexeerd wegschrijven van die waarde in het videogeheugen, adres $0578 + 0 = $0578.
Daarna de implied instructie INX, die de waarde in het x register met 1 verhoogt. Vervolgens springt hij terug naar adres $1007, achter de instructie om X te laden met 0, dit omdat anders het verhogen van X geen zin heeft.
Daarna gaat hij weer een getal geïndexeerd X laden, maar dit keer is het adres $2001, omdat het defaultadres $2000 is + de $01 van het X register wat in totaal $2001 is. Op deze wijze schrijft hij de waarde ook weer weg, en doet dit net zo vaak totdat hij het karakter 0 tegen komt bij de BEQ instructie, wat aan het einde van de string (JAN KLAASSEN) staat.
Dus niet vergeten om achter de hex waarde $4E de waarde $00 te zetten, want anders zal het programma het hele geheugen op je scherm printen net zolang tot hij een 0 tegen komt.

En verder....

Dit zijn alweer een paar instructies erbij waar je wat mee kunt experimenteren. Als het je nog niet helemaal duidelijk is, is het op dit moment nog te vroeg voor paniek, want we gaan de volgende keer nog wat dieper op dit alles in. Je kunt in ieder geval al wat spelen met de assembler. Dus alvast maar sterkte met de assembler en tot de volgende keer.

Tjipke van der Land

Misser !

In Commodore Info nr. 7 van afgelopen jaar (Jrg.4) zijn wat foutjes geslopen in het artikel over RESETten (pag.62).

Schrijver S. Hol stuurde ons een aantal opmerkingen hierover:

- Het woord 'connector' en 'uitgang' wordt door elkaar gebruikt, maar betekent hetzelfde in dit verband.
- De eerst genoemde opdracht in het artikel moet zijn:

POKE 2050,8 : SYS 42291 :
POKE 46,
PEEK (35)-(PEEK(781)253) :
POKE 45,
PEEK(781)+2 AND 255

De schrijver van dit artikel werkt bij de firma TurboSoftware(inc) te Oldenzaal.